SPECIALE DRIVE

COME RENDERE INCOPIABILI I TUOI DISCHI

LA MAPPA DI MEMORIA DEL 1541 COMMENTATA E IN ITALIANO

Tutto quello che vorresti sapere nell'unità disco del tuo C64/128

Systems

Commodore - N. 19 - Dicembre 1988 - Distr. MePe

SEMPRE PIU' IN ALTO

Non si può certo pretendere di esaurire, in poche decine di pagine, un argomento così complesso come lo studio di un disk drive.

La cosiddetta "architettura" del sistema, la connessione con un computer, la struttura delle Rom, la gestione della Ram sono, infatti, temi che offrono spunti sempre nuovi per applicazioni quasi sempre originali ed interessanti.

Il lettore, quindi, potrà restare deluso per la mancanza di alcune problematiche (e relative soluzioni) che lo hanno assillato per diverso tempo.

E' stata nostra intenzione approfondire ciò che non risulta chiaro sul libretto di istruzioni del versatile 1541; in alcuni casi, poi, è stata prestata particolare attenzione alla divulgazione di notizie non sempre facili da rintracciare.

I vari "capitoli" in cui è suddiviso il presente "speciale" dovrebbero contenere tutti gli ingredienti per usare il drive nel modo più sofisticato possibile.

Tuttavia ci rendiamo conto che l'argomento non può concludersi qui.

Come nostra abitudine, quindi, invitiamo il lettore a seguirci sulla rivista "Commodore Computer Club", distribuita mensilmente presso tutte le edicole, alla quale andranno indirizzate le domande, i dubbi, le richieste di chiarimenti, approfondimenti ed applicazioni da parte dei lettori.

Come nei precedenti fascicoli "speciali" ("Linguaggio Macchina" e "Totocalcio"), infatti, l'intervento dei nostri lettori ha sempre offerto lo spunto per affrontare nuovamente, o in modo diverso, tutti gli argomenti descritti nei fascicoli stessi.

DIRETTORE
Alessandro de Simone

CAPOREDATTORE
Michele Maggi

Hanno collaborato:
Alessandro Diano e Franco Rodella

UFFICIO GRAFICO:
Arturo Ciaglia, Elena Salvadori

EDIZIONI
Systems Editoriale s.r.l.
(Registro Nazionale Stampa n. 01500)

DIREZIONE, REDAZIONE, PUBBLICITA':
Via Mosè, 18 - 20090 Opera (MI)
Tel. (02) 5244125 - 5242743
Fax (02) 5244339 - Autorizzazione
del Tribunale di Milano n. 103 del 25/2/84
Direttore responsabile: Agostina Ronchetti

COMPOSIZIONI - FOTOLITO:
Systems Editoriale s.r.l.

STAMPA:
Stem - Milano

Concessionario esclusivo
per la diffusione MePe Spa
V.le Famagosta, 75 - Milano

SOMMARIO

INTRODUZIONE

Qualche tempo fa mi capitò di ascoltare le lamentele di un tizio a proposito della protezione dei programmi da parte delle ditte di software.

Egli le accusava di "oscurantismo" (testuale) in quanto...

"...se tanto i programmi si copiano lo stesso, perchè mai li proteggono? Forse non vogliono rivelare a tutti i loro segreti di programmazione".

Da un lato c'è da considerare l'inviolabile (non in Italia...) diritto alla tutela del lavoro di programmazione di un gruppo di persone tanto qualificate quanto costose, dall'altro la curiosità di molti utenti nei confronti del programma visto da "dentro", protezioni comprese.

Ed ecco il punto; talune scoperte di tipo "illegale", quali nuovi codici di istruzioni in L.M., gestioni parallele di I/O del drive e simili, sono conosciute in modo approfondito da tre categorie di persone: programmatori professionisti, operatori del settore (stampa compresa) ed "hackers".

In Italia il primo gruppo è alquanto sparuto, quasi prossimo all'inesistenza, probabilmente anche a causa della spaventosa prolificazione dei membri dell'ultimo gruppo le cui notizie "che servono davvero" vengono generalmente tenute "segrete" per un impiego personale, per pochi intimi, dei quali l'utente può solo ammirarne il risultato finale ad opera di fantasiosi schermi d'apertura, protezioni di programmi sprotetti(?!?) e simili "meraviglie" degne, a dire il vero, di miglior causa.

Ecco quindi che il povero utente si trova condannato all'ignoranza e si rifugia nelle riviste del settore le quali (ma non tutte), tentano di informarlo dei presunti misteri celati dagli "oscurantisti" programmatori di professione i quali, come ogni buon hacker che si rispetti, dopo aver speso Kilowatt davanti al computer per frugarne i meandri più reconditi, ben difficilmente li comunica al prossimo, in modo da stupire coloro i quali avranno tra le mani il risultato delle loro cognizioni...

Tra le lettere ultimamente arrivate in redazione, infatti, tale mancanza di informazioni è fortemente sentita: molti notano che tutti i loader degli attuali programmi commerciali impiegano routine che svolgono misteriose operazioni nella R.A.M. del 1541 e vorrebbero saperne di più. Un lettore ha chiesto, addirittura, un sistema per sproteggere i più famosi videogame e commentarne i disassemblati (Hey hackers, ci siete?) ed altri ancora vorrebbero approfondire il discorso sugli errori del disco: come si creano via software ma, soprattutto, come si eliminano.

Ecco, quindi, le ragioni di questo speciale Commodore sul disk drive 1541; da una parte, continuare il discorso "protezioni" già intrapreso da parecchio tempo sulla rivista "Commodore Computer Club", dall'altra portare l'utenza dell'ormai diffusa unità a disco ad un migliore livello di conoscenza della periferica, tramite tre serie di articoli e routine sia in assembly (commentato) sia in Basic, dedicate ai principianti, ai "normali" ed agli esperti, nelle quali potranno ovviamente essere inserite proprie modifiche in quanto, come tutti i prodotti dell'iniziativa "software made in Italy", l'assenza di protezioni, la presenza del supporto magnetico e, soprattutto, degli articoli esplicativi, li rende completamente disponibili alle personali sofisticazioni.

Infine, a conclusione dello "speciale", due complementi sicuramente graditi quali la completa ed unica mappa della memoria INTERAMENTE IN ITALIANO del drive 1541 ed un'idea per una protezione che rende realmente induplicabile un dischetto ad uso personale.

Rimanendo a completa disposizione per eventuali errori and / or omissioni, per concludere vorrei formulare un particolarmente doveroso ringraziamento all'ingegner de Simone, senza il quale questa pubblicazione avrebbe avuto serie difficoltà di parto...

Alessandro Diano

UN FENOMENO CHIAMATO 1541

"Iniziazione" alla più diffusa unità a dischi per i Commodore ad otto bit

Se, tra chi legge, qualche sfortunato ancora non possiede alcun disk-drive, probabilmente è disinformato su ciò che una tale unità può offrire in più rispetto all'obsoleto registratore a cassette.

Seppur con le sue pecche, infatti, una unità a dischi rappresenta uno dei midgliori investimenti per chi intenda fare un utilizzo serio del proprio calcolatore: attività di videoscrittura (Word-processing), di archiviazione dati (quali indirizzi, nomi, cioè Data - base) o di calcolo di tabelle e grafici (spreadsheet) e simili sono praticamente preclusi a chi pensa di archiviare qualche migliaio di record sul nastro magnetico di una cassetta.

Anche con un buon turbo-tape, infatti, il lavoro consisterebbe in un riavvolgimento continuo di nastro; a parte il rischio di cancellare inavvertitamente spezzoni di nastro di vitale importanza.

Tale scomodità è sostanzialmente dovuta al sistema di registrazione del datassette il quale è di tipo sequenziale; ciò signfica che la scrittura dell'informazione (byte di programma, dato generico od altro) avviene inviando alla testina di scrittura un gruppo di valori, l'uno dietro l'altro, depositati in fila sulla superficie del supporto magnetico.

La procedura prevede che la lettura avvenga nella stessa maniera, e cioè che l'ultimo dato venga letto SOLO dopo che TUTTI i dati precedenti, interessanti o meno, sono transitati davanti alla testina di lettura / scrittura.

Uno dei più notevoli vantaggi dell'unità a dischi è proprio dato dal sistema di registrazione delle informazioni, che avviene su una superficie magnetica pressochè identica a quella del nastro delle cassette, ma con sostanzile differenza nella forma: il dischetto è racchiuso in una custodia protettiva che può essere del tipo rigido, oppure flessibile.

Il 1541 si serve, per la conservazione dei dati, di quest'ultimo tipo e precisamente della categoria universalmente nota come "5 pollici ed 1/4" (oppure 5.25), nome derivato da una delle unità di misura di lunghezza anglosassoni: 5.25 pollici x 2.54 centimetri (misura in cm. di un pollice) = 13.34 centimetri, che è, appunto, la misura del lato della custodia quadrata del dischetto.

Questo è costituito da un supporto di mylar flessibile sul quale viene letteralmente "spalmato" l'ossido magnetico presente nei comunissimi nastri in cassetta ed in bobina, il quale verrà opportunamente magnetizzato da un'apposita testina per la conservazione delle informazioni.

Prima di qualunque operazione di scrittura e/o di lettura, però, è necessario dare un nome di (massimo) 16 caratteri, ed un identificatore di due, al dischetto stesso, il quale (a differenza della cassetta, che non ne ha bisogno) deve essere diviso in varie zone concentriche (in tutto ve ne sono 35, dette "traccie") ciascuna delle quali viene ulteriormente frazionata in sotto-zone più piccole denominate settori; all'inizio di ciascuno di essi, inoltre, viene scritta un'intestazione che, oltre a varie somme di controllo (dette checksum), contiene le indicazioni di traccia e settore, ovviamente differenti da zona a zona, che consentiranno, in seguito, l'univoca determinazione della parte del disco nella quale ci si trova.

Durante l'operazione di preparazione del dischetto (effettuata su una sola delle due facce, essendo il 1541 un drive, appunto, a singola faccia), comunemente denominata "formattazione", viene anche preparato lo spazio per ospitare un indice (vuoto, all'inizio) destinato a contenere l'elenco di tutto quanto è registrato sul dischetto stesso (caratteristica inesistente nel datassette), che consente una più facile ricerca ed identificazione di uno specifico "file", termine con il quale si indica un qualunque insieme di dati registrato sul disco.

Generalmente i file maggiormente usati sono quelli di tipo programma (che il disk drive 1541 chiama PRG), ma ve ne sono anche altri tipi dei quali si specificherà in seguito.

Relizzato l'indice, pronto da riempire (detto "directory"), viene quindi creata una sorta di mappa che, per ciascun settore di ciascuna traccia, consente di stabilire se lo stesso è già stato occupato da altri dati precedenti (e non è quindi disponibile per la registrazione di quelli attuali), oppure se risulta libero di essere eventualmente riempito con i dati in arrivo o con le successive registrazioni; tale tabella è, quindi, una mappa che indica la disponibilità (o meno) dei settori (chiamati anche blocchi) del disco: in inglese si chiama Block Availability Map, comunemente abbreviato nella sigla B.A.M.

Dal momento che queste due importanti indicazioni dovranno essere consultate spesso dal drive per evitare di cercare sul disco un file che non è presente neppure nell'indice oppure, peggio, di scrivere un nuovo file sopra ad uno già esistente, i bravi personaggi che hanno concepito l'unità a dischi hanno pensato di mettere le informazioni stesse a portata di mano, sistemando la B.A.M. nel settore 0 della traccia 18, intelligentemente posizionata a metà strada tra le due traccie più lontane.

La traccia n. 1 è la più esterna, e la 35ma, invece, la più interna; la directory, ossia l'elenco di tutti i file del dischetto, occupa il settore 1 della traccia 18 e, in caso di riempimento dell'indice in questo settore, la directory verrà fatta "traboccare" nei blocchi successivi alla diciottesima traccia.

La lettura e scrittura dei dati, quindi, viene gestita da un apposito sistema operativo presente nel drive (il Disk Operating System, detto anche D.O.S.) che nel 1541 è implementato nella versione 2.6; ciò consente al 1541 di vantare la denominazione di "periferica intelligente" in quanto, a differenza del registratore a cassette (totalmente gestito dal computer), il disk drive possiede una propria R.O.M. (il D.O.S. appunto), una R.A.M. e addirittura un microprocessore che consentono un trattamento più efficace delle informazioni provenienti da (e "viaggianti" verso) il computer.

Cercando un dato file, il D.O.S. non si leg-

gerà tutti quelli precedenti per ricavare quello scelto, bensì andrà a consultare la directory dalla quale avrà l'indicazione immediata della traccia e del settore dai quali inizia il file richiesto.

Quindi, in relazione alla sua posizione attuale (diciottesima traccia) la testina dell'unità a dischi si sposterà, proprio come il braccio di un giradischi musicale, di un numero di scatti gestiti da un motore passo-passo (stepper motor) il quale provvederà a far raggiungere la traccia richiesta.

Ivi posizionatasi la testina, avrà inizio la ricerca del "SYNC" ossia del carattere di sincronismo che indica, da quel punto in poi, l'inizio dei dati dell'intestazione del blocco (in inglese: block header); letta quest'ultima si verificherà se corrisponde al blocco cercato e, in caso affermativo, si inizierà la lettura o la scrittura del file.

Ciascuno dei blocchi del disco è normalmente in grado di contenere un massimo di 254 byte su 256 disponibili (un settore = un blocco = 256 byte immagazzinati sul dischetto); i primi due byte di ogni settore, infatti, contengono due indicazioni, rispettivamente di traccia e settore, che individuano il blocco "concatenato" a quello presente, nel caso che il file sia più lungo di 254 byte; in altri termini, letti i byte dal 2 al 255 il D.O.S. si recherà nella traccia contenuta nel byte 0, al settore contenuto nel byte 1, per proseguire la lettura del file.

Le due indicazioni sono gestite automaticamente dal D.O.S. che, al momento della registrazione del file, provvede a scriverle correttamente, concatenando tutti i blocchi nei quali è contenuto un dato file.

Nel caso quest'ultimo sia lungo meno di 255 byte, oppure ci si trovi nell'ultimo settore del file, i primi due byte (0 ed 1) del blocco conterranno il valore 0 (infatti non esiste nessuna traccia 0) ed il numero dei byte occupati nel presente settore: il complemento a 255 non verrà considerato.

STRUTTURA DEL DISCO

Si è visto che il D.O.S. esegue la formattazione del dischetto suddividendolo in 35 traccie concentriche ciascuna delle quali suddivisa in un imprecisato numero di settori; dal momento che la traccia più interna (la 35) ha una superficie minore di quella più esterna (la 1), il numero di settori diminuirà a mano a mano che, dalle traccie esterne, si procede verso quelle interne (vedi figura 1).

Con un rapido calcolo si trova che il numero totale dei settori presenti sul dischetto è 683, valore che, moltiplicato per la capacità di ciascun settore (256 byte), fornisce una capacità teorica di 174848 byte, ossia di 170.75 Kilobyte.

```
--------------- FIGURA 1 --------------
N.ro della traccia     Numero di settori

 dalla  1 alla 17        21 (da 0 a 20)
 dalla 18 alla 24        19 (da 0 a 18)
 dalla 25 alla 30        18 (da 0 a 17)
 dalla 31 alla 35        17 (da 0 a 16)

----------------------------------------
```

```
--------------- FIGURA 2 --------------
TRACCIA 18, SETTORE 0

              Contenuto
N.ro byte       (esa)     Indicazioni

   0            $12       Traccia 18
   1            $01       Settore 1 (Directory
   2            $41       Codice ASCII della lettera "A"; formato di scrittura
                          del 1541
   3            $00       Non usato; nel 1571, drive C.B.M. a doppia faccia, un
                          valore di $80 identifica un dischetto
                          in tale modalita'.
da 4 a 143      $??       35 gruppi di 4 bytes di indicazioni blocco libero/
                          occupato per ciascuna traccia
da 144 a 159    $??       Nome dato al dischetto al momento della formattazione;
                          se inferiore ai 16 caratteri i rimanenti contengono
                          il codice $A0 dello spazio shiftato
da 160 a 161    $A0       Codice ASCII dello spazio shiftato usato come
                          separatore tra il nome del dischetto e la sua Id.
                          (i due caratteri di identificazione) durante la
                          visualizzazione della directory
   162          $??       Primo dei due valori di identificazione (Id.) dati al
                          dischetto al momento della formattazione
   163          $??       Secondo dei due valori di identificazione (Id.) dati
                          al dischetto al momento della formattazione
   164          $A0       Byte inutilizzato: codice ASCII dello spazio shiftato
   165          $32       Codice ASCII del numero "2"
   166          $41       Codice ASCII della lettera "A" che, insieme al byte
                          precedente forma l'indicazione identificativa del
                          1541 "2A" che il D.O.S. aggiunge a ciascun dischetto
                          formattato per suo tramite
da 167 a 170    $A0       Bytes inutilizzati: codici ASCII dello spazio shiftato
da 171 a 255    $00       Bytes inutilizzati
----------------------------------------
```

```
------------- REM FIGURA 3 ------------
N.ro del byte    Indicazioni

     0           Numero dei blocchi liberi disponibili nella traccia
     1           Byte i cui otto bit si riferiscono ai settori dallo 0 al 7
     2           Byte i cui otto bit si riferiscono ai settori dall' 8 al 15
     3           Byte i cui otto bit si riferiscono ai settori dal 16
                 all'ultimo della traccia; gli eventuali bits eccedenti
                 sono inutilizzati
----------------------------------------
```

```
------------- REM FIGURA 4 ------------
TRACCIA 18  SETTORE 1

N.ro di byte  Indicazioni

    0         Traccia del blocco successivo della directory
              (nell'ultimo blocco contiene $00 esa = 0 decimale)
    1         Settore del blocco successivo della directory
              (nell'ultimo blocco contiene $FF esa = 255 decimale)
da 2 a 31     30 bytes di dati del primo file della directory
da 32 a 33    2 bytes inutilizzati contenenti $00
da 34 a 63    30 bytes di dati del secondo file della directory
da 64 a 65    2 bytes inutilizzati contenenti $00
da 66 a 95    30 bytes di dati del terzo file della directory
da 96 a 97    2 bytes inutilizzati contenenti $00
da 98 a 127   30 bytes di dati del quarto file della directory
da 128 a 129  2 bytes inutilizzati contenenti $00
da 130 a 159  30 bytes di dati del quinto file della directory
da 160 a 161  2 bytes inutilizzati contenenti $00
da 162 a 191  30 bytes di dati del sesto file della directory
da 192 a 193  2 bytes inutilizzati contenenti $00
da 194 a 223  30 bytes di dati del settimo file della directory
da 224 a 225  2 bytes inutilizzati contenenti $00
da 226 a 255  30 bytes di dati dell' ottavo file della directory
----------------------------------------
```

In pratica, però, l'intera traccia 18 è riservata alle informazioni sul dischetto gestite dal D.O.S. e, pertanto, i blocchi realmente disponibili sono 683 - 19 = 664. Tale è, infatti, il valore che si ottiene chiedendo la directory (LOAD "$",8 e LIST) subito dopo la formattazione di un dischetto.

Dal canto suo la B.A.M. provvede a rendere non disponibili tutti i settori della diciottesima traccia, e configura i valori del settore 0 (la B.A.M.) come da figura 2.

Come si nota, nei byte dal 4 al 143 vi sono 140 valori divisi in 35 gruppi di 4 byte ciascuno (35 x 4 = 140 byte): in figura 3 è rappresentato l'esatto significato dei 4 valori che è possibile trovare, tenendo presente che, a ciascun settore del dischetto, è associato un singolo bit il cui stato logico indica l'avvenuta occupazione, o meno, del blocco al quale si riferisce. La simbologia è intuitiva:
bit posto ad 1 = settore libero
bit posto a 0 = settore occupato

Ad esempio un byte contenente $FF (255 decimale) con i suoi otto bit posti al valore unitario, indica che gli otto blocchi al quale si riferisce sono tutti liberi.

LA DIRECTORY

La Directory, che contribuisce a rendere il drive così diverso dal datassette, è costituita da un elenco di tutto quanto è presente sul dischetto e da un folto gruppo di varie informazioni, quali:

- Tipo di ciascuno dei file presenti
- Numero di traccia e di settore, per ciascun file, del primo blocco dati; quelli successivi sono concatenati, come già visto, grazie ai byte 0 ed 1 di ciascun settore
- Nome dei file presenti
- Numero dei blocchi occupati da ciascun file

Trasferendosi ora nel settore 1 della traccia 18 (dal quale ha inizio la directory), ed osservando la figura 4, si può comprendere meglio come il D.O.S. memorizza le informazioni sui file.

In pratica viene creato un gruppo di trenta byte per ciascuno dei file presenti i quali, in un solo settore, possono essere contenuti nel numero massimo di otto. Sapendo che la traccia 18 comprende in totale 19 settori (numerati da 0 a 18) e che il primo di essi (il n. 0) è occupato dalla B.A.M., si può rapidamente determinare il massimo spazio consentito alla directory di un dischetto:

18 sett. x 8 file per sett. = 144 file

...valore che rappresenta il numero massimo di file memorizzabili in un singolo floppy disk. I 30 byte di dati, che caratterizzano ognuno di questi, sono suddivisi come indicato in figura 5.

Gli ultimi due byte di ciascun "data entry" (il gruppo di trenta byte dedicati ad ogni file della directory) contiene lo spazio, in numero di settori, occupato dal file secondo il formato classico (per chi lo conosce...) low / high ossia basso / alto.

Esso prevede che il numero dei blocchi sia dato dal seguente valore:

(High x 256) + (Low) = Blocchi occupati

IL TIPO DI FILE

Si è visto come viene organizzata la tabella dei dati di un file e come il primo di tali valori indichi il tipo di file registrato.

Questo, però, comprende anche l'indicazione di altri "stati" nei quali si può trovare uno dei possibili tipi di file i quali, in tutto, sono cinque:

- Tipo REL (Relativo)

Sono file gestiti da apposite routine del D.O.S. con i quali è possibile la creazione di archivi di dati suddivisi in vari record (es. nome, cognome e telefono) velocemente rintracciabili fornendo il solo numero di record. Tali tipi di file vengono generalmente creati da programmi di archiviazione dati (data - base e simili) e sono preferibilmente gestibili tramite gli stessi piuttosto che direttamente dall'utente.

- Tipo USR (User, cioè utente)

Sono file contenenti dati non organizzati dal D.O.S. (come nel caso dei REL) ma ad uso e consumo dell'utente per altri scopi che, di solito, non sono l'archiviazione di record, programmi o sequenze di dati ordinati. Come i REL, sebbene creabili direttamente, sono generalmente adoperati da programmi che ne consentono la creazione, l'uso e l'eventuale cancellazione.

Gli impieghi tipici per tale tipo di file comprendono i comandi di utility loader (dei quali si tratterà ampiamente in seguito) ed i file "vuoti" dai nomi formati da caratteri grafici (segni di "meno", crocette etc.) adoperati come separatori tra i nomi di file "veri" della directory.

- Tipo PRG (Programma)

Questo tipo di file non ha bisogno di presentazioni in quanto è il tipo di file maggiormente usato: comprende i codici (basic e / o assembly) di programmi direttamente eseguibili dall'interprete Basic o dal microprocessore 6510.

- Tipo SEQ (Sequenziale)

Questa categoria comprende una sequenza di dati che ha senso considerare come tale: generalmente l'applicazione tipica prevede dei file di testo (come quelli generati da un word - processor) oppure generici output da riversare su carta.

- Tipo DEL (Delete, cioè cancellato)

E' un tipo di file fittizio in quanto cancel-

```
-------------- REM FIGURA 5 ------------
DATA ENTRY DI UN FILE 1541

N.ro byte   Indicazioni

    0       Tipo di file
    1       Traccia del primo blocco di dati del file
    2       Settore del primo blocco di dati del file
da 3 a 18   Nome del file; in caso di lunghezza inferiore ai 16 caratteri,
            i bytes inutilizzati contengono $A0 (spazio shiftato)
   19       (Solo per i files di tipo REL) Traccia del primo blocco di
            side-sector
   20       (Solo per i files di tipo REL) Settore del primo blocco di
            side-sector
   21       (Solo per i files di tipo REL) Lunghezza del record
da 22 a 25  Bytes inutilizzati contenenti $00
   26       (Solo utilizzando il comando di "Chiocciolina" per la
            riscrittura di un file) Traccia del nuovo file
   27       (Solo utilizzando il comando di "Chiocciolina" per la
            riscrittura di un file) Settore del nuovo file
   28       Numero di blocchi occupati dal file (low)
   29       Numero di blocchi occupati dal file (high)
----------------------------------------
```

lando normalmente un file, non appare nella directory.

Sostanzialmente individua tutti quei file (che una volta erano REL, USR, PRG oppure SEQ) che sono stati cancellati con un apposito comando di Scratch previsto dal D.O.S. e non figurano più come disponibili nella directory; la dicitura "DEL" nella directory, appare solamente modificando opportunamente il byte del tipo di file dopo il comando di Scratch (come vedremo).

Si parlava di altri stati nei quali possono venire a trovarsi i vari tipi di file che, nel 1541, appartengono a due tipologie, le quali possono anche coesistere:

1. File protetto dalla cancellazione (oppure no)
2. File lasciato "aperto" oppure chiuso correttamente

Il primo caso identifica una proprietà che possono avere tutti i file (persino i DEL!) di "protezione" dallo Scratch, con la quale l'omonimo comando non è più impiegabile per cancellare un file dalla directory.

Se tale proprietà è impiegata (file protetto), nella directory, subito dopo il tipo di file, verrà visualizzato il segno di minore (<).

Il secondo caso di file "aperto" si può verificare quando, una volta iniziata la scrittura di un file, non si chiuda correttamente la trasmissione dei dati con l'apposito comando CLOSE oppure, ad esempio, si spenga il disk drive durante un comando di SAVE (meglio però non provare a farlo...).

In caso, quindi, il file rimanga aperto, la B.A.M. non sarà a conoscenza di quanti e quali blocchi occupa il file (con tutti i relativi rischi di indesiderate sovrascritture, come ben si può immaginare), i settori nei quali è contenuto il file non sono collegati correttamente tra loro (si vedano i byte 0 ed 1 di ciascun settore) e l'indicazione del tipo di file nella directory sarà preceduta da un asterisco (*) ad indicare l'anomala condizione del file.

```
-------------- REM FIGURA 6 ------------

BYTE DI INDICAZIONE DEL TIPO DI FILE

Bit  Stato 1            Stato 0

 7   File chiuso        File aperto
 6   File protetto      File cancellabile
 5   File riscritto     File nuovo
 4           Non utilizzato
 3           Non utilizzato
 2            Tipo di file
 1            Tipo di file
 0            Tipo di file

-----------------------------------------
```

Come regola generale, si tenga presente che, in questi casi, è opportuno eliminare i file così conciati con un comando di Validate anzichè con uno di Scratch (di entrambi se ne riparlerà in seguito), in quanto quest'ultimo libererà nella B.A.M. tutti quei settori concatenati al primo del file "aperto" il quale, siccome non ha i due byte di collegamento corretti con gli altri settori (infatti non è stato "chiuso" correttamente), rischia di far cancellare anche i settori occupati da altri file "buoni" creando una situazione risolvibile solo da un comando di Validate se, ovviamente, nel frattempo non si è scritto nient'altro sul dischetto.

Il byte di indicazione del tipo di file, come tutti i byte, è formato da 8 bit dai quali si possono dedurre tutte le informazioni sopra descritte (vedi figura 6).

Per quanto riguarda i tre bit (dallo 0 al 2) del tipo di file vero e proprio, si considerino le seguenti sequenze binarie riferite, da sinistra a destra, dal bit 2 al bit 0 (cioè 2, 1 e 0):

000 = DEL
001 = SEQ
010 = PRG
011 = USR
100 = REL

E NON E' TUTTO...

Con questo capitolo si è cercato di fornire una prima infarinatura per consentire, in seguito, l'uso di una terminologia comprensibile tanto dai "maghi" quanto dai neofiti.

Approfondiremo, in un secondo momento, le caratteristiche del disk drive 1541 facendo continuamente riferimento al presente capitolo, che supporremo completamente acquisito.

Mentre l'importanza dei bit 7 e 6 dovrebbe risultare evidente, il bit 5, e quelli numerati da 0 a 2, meritano un approfondimento.

Nel capitolo dedicato ai comandi riconoscibili dal disk drive 1541 si vedrà che ne esiste uno formato dal carattere di chiocciolina (tasto compreso tra l'asterisco ed il carattere "P"), il quale, digitato davanti al nome del file durante un'operazione di SAVE, comunica al D.O.S. che, prima di registrare il file in arrivo, dovrà cancellare un file già esistente, dotato dello stesso nome e, quindi, sovrascriverlo con il nuovo file.

Durante questa operazione, il 1541 "marchia" l'indicatore del tipo del file in oggetto, settando (cioè mettendo ad uno) il bit 5 dello stesso.

COME SI PARLA AL DISK DRIVE 1541

Guida pratica alla comunicazione in Basic tra computer e drive per la gestione completa del floppy-disk

Risulta strano come la maggioranza dell'utenza del 1541 lo consideri (e lo adoperi) esclusivamente come un registra-programmi più veloce (anzi: meno lento) del datassette.

Il più grosso problema è sicuramente la mancanza di informazioni al riguardo che, molto spesso, non vanno al di là di quanto può offrire il semplice libretto di istruzioni, lasciato, peraltro, in un penoso stato di emarginazione totale dalla gran parte degli utilizzatori del drive.

Premesso che l'attenta lettura del libretto, da sola, fornisce almeno sei mesi di esperienza in più (e di perdita di tempo in meno), si vedrà come impiegare correttamente i vari comandi (e relative opzioni) riconoscibili dal 1541.

UN PO' DI BASIC

La comunicazione tra il computer ed il disk drive avviene attraverso un'apposita porta (il cosiddetto bus seriale) presente su entrambi i dispositivi; onde consentire la gestione di più comunicazioni in contemporanea (ad esempio due di lettura e tre di scrittura), il "protocollo" dello scambio di informazioni prevede che, a ciascuna comunicazione, venga assegnato un ben preciso "canale" (in sostanza un numero) al quale si dovrà fare riferimento per tutte le operazioni successive.

Tale assegnazione viene fatta all'atto dell'apertura del file, ossia quando si specifica l'operazione da fare (invio, oppure ricezione di dati) che, in caso di comandi dedicati, viene gestita totalmente dal computer.

Un esempio di comando Basic dedicato è LOAD oppure SAVE: non vi è nessuna specifica di canale da fare, in quanto viene scelto quello idoneo all'operazione richiesta: per il LOAD si usa il canale 0, mentre per il SAVE il canale 1, anche se la faccenda non interessa assolutamente all'utente in quanto è il computer che si occupa di selezionare l'apertura del canale corretto, 0 ed 1 nell'esempio riportato.

Se, però, si desidera operare direttamente su un file, l'apertura (ed i problemi relativi) del canale è nelle mani dell'utente al quale, comunque, il linguaggio Basic e le routine del Kernal (R.O.M. del computer predisposta all'esecuzione di alcune funzioni fondamentali tra le quali la ricezione e l'invio dei dati da e per il bus seriale) forniscono aiuto.

Il comando Basic necessario per iniziare una comunicazione è OPEN, solitamente seguito da alcuni parametri che possono avere vari significati:

OPEN x, y, z, "stringa"

Il numero logico "X" del file può variare tra 0 e 255; serve al computer per distinguere, tra i vari file (eventualmente) aperti in contemporanea, quale adoperare per l'operazione di invio oppure di ricezione dati.

E' comunque preferibile adoperare i numeri di file logico compresi tra 1 e 127 in quanto il numero zero genera un punto interrogativo in fase di ingresso dati (come l'istruzione Basic INPUT) mentre i valori da 128 a 255 possono generare, in seguito all'invio dei dati, un carattere di "a capo" dopo quello di ritorno normale, con il rischio, specie negli output su stampante, di formati di uscita non desiderati.

Il secondo numero da specificare (Y) in un'istruzione di OPEN rappresenta il dispositivo periferico con il quale si desidera scambiare informazioni (figura 7) e, se maggiore di 3, identifica una periferica collegata al bus seriale; solitamente, per il disk, drive assume il valore di 8.

Il parametro "Z" è il più importante del comando anche se, a differenza dei due parametri precedenti, non è obbligatorio (il default, cioè il valore assunto come standard se non si specifica altrimenti, è lo 0).

Per il disk drive esiste la possibilità di gestire sino ad un massimo 16 canali numerati da 0 a 15 (figura 8).

Il numero del canale, quindi, (detto anche "indirizzo secondario"), serve per comunicare alla periferica "Y" (con la quale si è aperto il file logico numero "X"), quale operazione si vuole eseguire tra il LOAD, il SAVE, la gestione di file dati e l'invio di comandi al 1541.

La "Stringa" di OPEN rappresenta una stringa di caratteri (opzionale) la quale, oltre a specificare l'eventuale nome del file, definisce ulteriori modalità di trattazione del file in oggetto.

Ecco ora qualche esempio di OPEN:

OPEN 1, 8, 0, "$"

Apre il file logico numero 1, sulla periferica 8 (il disk drive) per un LOAD (indirizzo secondario = 0) del file chiamato "$" (la directory)

OPEN 127, 8, 15

Apre il file logico numero 127, sulla periferica 8 (il disk drive) per un invio di comandi (indirizzo secondario = 15)

OPEN 10, 11, 12, "PIPPO, S, R"

Apre il file logico numero 10, sulla periferica 11 (l'eventuale quarto disk drive) per una gestione del file dati "PIPPO" (indirizzo secondario = 12 cioè compreso tra 2 e 14)

Mentre negli esempi precedenti era l'indirizzo secondario (o numero del canale) che determinava l'operazione richiesta (lettura, scrittura oppure altro) in quest'ultima OPEN si è impiegato il canale 12 che risulta perfettamente identico a quelli compresi tra 2 e 14, i quali non specificano ciò che si vuole fare sul file ("PIPPO" nell'esempio); quest'ultima indicazione, infatti, viene riportata nel nome della OPEN con due lettere separate da altrettante virgole.

Nell'esempio, la stringa (cioè l'insieme di caratteri) "PIPPO, S, R" significa che sul file "PIPPO", che è un file di tipo SEQ (lo specifica la lettera "S"), si vuole effettuare un'o-

perazione di lettura (lettera "R" = Read = Lettura).

Chiaramente, se il file PIPPO non è del tipo SEQ, la lettera "S" può essere sostituita dalle seguenti:

D = DEL
S = SEQ
P = PRG
U = USR

Il primo tipo (DEL), sebbene previsto nella tavola delle iniziali dei tipi di file del 1541, è preferibilmente da non utilizzare in quanto, essendo il DEL una tipologia "illegale", la gestione dello stesso può causare sorprese poco simpatiche.

Si sarà notato che manca l'iniziale del tipo di file REL la quale, infatti, va descritta a parte.

Mentre le lettere D, S, P ed U devono essere seguite (oltre che dalla virgola di separazione) dalla modalità desiderata ("R" = Read = Lettura nell'esempio del file PIPPO), la lettera del tipo REL (che è una "L") deve essere seguita da un numero (sotto forma di CHR$) che rappresenta la lunghezza del file relativo al momento della creazione; esempio:

OPEN 2, 8, 2, "PIPPO, L, " + CHR$ (100)

Apre un file relativo avente una lunghezza di 100 caratteri per ciascun record.

Sempre al posto dell'iniziale del file (D, S, P, U oppure L), infine, può anche trovarsi la lettera "A" (Append = aggiungere) la quale, normalmente, viene impiegata per aggiungere una nuova lista di dati ad un file SEQ già presente sul dischetto.

Per far ciò è sufficiente riaprire il file SEQ (nell'esempio chiamato PIPPO) con la seguente sintassi...

OPEN 3, 8, 3, "PIPPO, A"

...e quindi scrivere la nuova lista dati con le modalità descritte in seguito.

Passando a descrivere le varianti della lettera "R" di Read, si trovano tre distinte possibilità:

R = Read = Lettura
W = Write = Scrittura
M = Modify = Modifica

Il primo tipo informa il D.O.S. che si leggeranno dati dal file specificato; analogamente la "W" indicherà, invece, un'operazione di scrittura nel file che si sta creando, mentre la "M" è l'ultima possibilità con la quale si può tentare di leggere un file di dati lasciato "aperto" (asterisco "*" prima dell'indicazione del tipo di file nella directory) e riversare gli stessi dati in un nuovo file in modalità "W"rite, tenendo però presente che i file non correttamente chiusi, in quanto tali, non hanno una fine corretta e, pertanto, è consigliabile analizzare carattere per carattere prima di scrivere nel nuovo file.

Ad ogni modo non ci si preoccupi troppo, per ora, di comprendere i concetti di file relativi, di come si fa a leggere carattere per carattere e così via, in quanto verranno trattati esaurientemente in seguito. A questo punto importa aver compreso il significato di tutte le lettere riservate, del tipo di file (D, S, P, U ed L) e di modalità operativa (R, W, M oppure A); nient'altro.

Ecco ora alcuni esempi di quanto descritto:

OPEN 4, 8, 4, "PLUTO, P, W"

Apre in scrittura un nuovo file di tipo programma chiamato PLUTO.

OPEN 5, 9, 11, "DUMMY, U, M"
OPEN 7, 8, 10, "MINNIE, S, W"

Apre con il numero di file logico 5, sulla periferica numero 9 (secondo drive) ed in-

```
-------------- REM FIGURA 7 ------------

NUMERI DISTINTIVI DEI DISPOSITIVI PERIFERICI
Numero        Dispositivo periferico

  0           Tastiera
  1           Registratore a cassette
  2           RS 232
  3           Schermo
  4           Stampante
  5           Seconda stampante
  6           Plotter
  7           Secondo plotter
  8           Disk drive
  9           Secondo disk drive
 10           Terzo disk drive
 11           Quarto disk drive
da 12 a 255   Altri valori disponibili
              per future espansioni

----------------------------------------


------------- REM FIGURA 8 ------------

N.ro del canale  Impiego

     0           LOAD
     1           SAVE
da 2 a 14        File di dati sia
                 in lettura sia in
                 scrittura
    15           Canale di comando

----------------------------------------


------------- REM FIGURA 9 ------------

SIGNIFICATI DELLA VARIABILE RISERVATA ST

Periferica   Bit Valore Tipo di errore

Datassette    7   -128  Termine del nastro
              6    64   Termine del file
              5    32   Errore di checksum
              4    16   Errore di lettura

Porta seriale 7   -128  Periferica non presente
              6    64   Termine del file/termine della linea
              1     2   Ritardo trascorso in lettura
              0     1   Ritardo trascorso in scrittura

----------------------------------------


------------ REM FIGURA 10 -------------

PROGRAMMA BASIC DI CONTROLLO DELL'ERRORE NUMERO 5 (?DEVICE NOT PRESENT)

100 REM VERSIONE IN BASIC V2.0
110 DV=8: GOSUB 190: REM 8 = DRIVE
120 IF FL=1 THEN 150
130 PRINT "PERIFERICA PRESENTE!"
140 END: REM GOTO [INIZIO PRG.]
150 PRINT "PERIFERICA ASSENTE!"
160 END: REM GOTO [ERRORE]
170 :
180 REM SUBROUTINE DI CONTROLLO
190 OPEN 255,DV,15: CLOSE 255
200 FL=0: IF ST=-128 THEN FL=1
210 RETURN
----------------------------------------
```

dirizzo secondario 11, il file DUMMY di tipo USR per una modifica (tentativo di recupero) e prepara un'altra apertura sul drive numero 8, con numero logico di file 7 sul canale 10, di un file SEQ di "salvataggio" (in modalità di scrittura) chiamato MINNIE.

IL TRANSITO DEI DATI DA E PER IL 1541

A questo punto si è in grado di aprire correttamente un file di dati di qualsiasi tipo, ma non si hanno ancora tutti gli strumenti per scambiare i dati veri e propri, siano essi in ingresso (input) oppure in uscita (output).

Per le due direzioni possibili sono stati previsti tre comandi differenti, due di input ed uno di output:

GET#
INPUT#
PRINT#

La caratteristica distintiva dei tre comandi è quella dell'ultimo carattere di diesis (#) detto anche "cancelletto" dopo il quale deve essere indicato il numero logico del file al quale ci si vuole riferire.

Comando GET#

GET# [numero file] , [lista varibili separate da virgole]

Questo comando di input è di gran lunga il più usato in quanto consente di ricevere, dal file, un carattere per volta con il grande vantaggio di poterlo analizzare prima di inserirlo in qualche variabile "definitiva" oppure adoperarlo in generale.

La lista variabili che segue il numero di file può essere di qualunque genere, anche se è preferibile impiegare le variabili di tipo stringa per meglio analizzarle ed, eventualmente, estrarne il corrispondente valore numerico con l'istruzione Basic VAL; esempio:

```
OPEN 126, 8, 2, "CIAO, P, R":
GET# 126, A$, B$
```

Legge i primi due caratteri del programma CIAO e li deposita nelle variabili A$ e B$.

Comando INPUT#

INPUT# [numero file] , [lista varibili separate da virgole]

Questo comando è differente dal precedente in quanto legge un gruppo di caratteri alla volta, sino a quando, cioè, non incontra un crattere di separazione valido quale il return, CHR$(13), il punto e virgola, CHR$(59), oppure la virgola, CHR$(44): esempio

```
OPEN 99, 8, 14, "BABAU, S, R":
INPUT# 99, C$, C$, D$, D$
```

Legge i caratteri sino al quarto separatore del file SEQ chiamato BABAU. Al termine del comando nelle variabili C$ e D$ vi saranno, rispettivamente, il secondo ed il quarto gruppo letto.

Comando PRINT#

PRINT# [numero file] , [lista da inviare]

Questo è l'unico comando di output ed il suo utilizzo è pressochè analogo all'istruzione PRINT tradizionale; la differenza risiede nel fatto che la lista da inviare verrà indirizzata alla periferica associata nell'OPEN al numerò di file logico di PRINT#; esempio:

```
10 OPEN 77, 8, 13, "FIRULI', S, A"
20 PRINT# 77, "TELEFONO"; CHR$(13);
30 PRINT# 77, "CASA"
40 PRINT# 77, E$; CHR$(13); F$; G$
```

Apre il file SEQ chiamato FIRULI' per aggiungervi in coda ulteriori dati. Scrive quindi la stringa "TELEFONO", un carattere di Return di separazione, "CASA", la variabile E$, un altro carattere di separazione ed infine le variabili F$ e G$ le quali, in fase di lettura, costituiranno un solo gruppo (notare il carattere di punto e virgola).

Si noti, quindi, come i separatori possano influenzare il numero delle variabili necessarie alla rilettura di un file: per mantenere separate F$ e G$ si può inserire un CHR$(13), un CHR$(59), oppure un CHR$(44) anche se è preferibile adoperare sempre il CHR$(13) come unico carattere di separazione visto che il computer lo aggiunge normalmente al termine di una stringa; a tal proposito si noti come la linea 20 e la 30 risultino identiche, in quanto il CHR$(13); (comprensivo di punto e virgola) della linea 20 viene automaticamente creato dal calcolatore nella 30 subito dopo "CASA", visto che non vi è alcun separatore:

```
PRINT# 77, CHR$(13);: REM CORRETTO MA INUTILE
```

Genera un solo Return

```
PRINT# 77, CHR$(13): REM DA NON USARE!
```

Genera un doppio Return: uno del CHR$ (specificato esplicitamente) ed uno generato dal sistema a causa dell'assenza del punto e virgola

```
PRINT# 77: REM CORRETTO
```

Genera un solo Return (linea vuota)

CLOSE E STATUS

Oramai si sa quasi tutto su ciò che il Basic mette a disposizione per l'utilizzo delle preziose risorse di un disk drive 1541; resta da vedere l'ultima istruzione adoperata dopo una comunicazione Input / Output (I/O) tra calcolatore e periferica numero 8: CLOSE.

In inglese (lingua madre dell'informatica in generale) significa "chiudi" (così come OPEN significa "apri") ed in Basic svolge l'omonima funzione di chiusura di un file, semplicemente specificando il numero di file logico; esempio:

```
OPEN 5, 9: CLOSE 5
```

Apre un canale di comunicazione con il drive 9 e numero di file logico 5 che chiude subito dopo: in pratica non fa nulla.

Dopo la semplicissima sintassi di CLOSE si può ora introdurre uno degli ausili più potenti che il computer mette a disposizione del programmatore di routine I/O: la variabile riservata ST.

Si tratta di una variabile ad uso e consumo esclusivo del computer che non può essere assegnata dall'utente: qualsiasi tentativo di impostare ST uguale a qualcosa, si infrangerà contro l'inevitabile ?SYNTAX ERROR.

ST è l'abbreviazione di STATUS e contiene un valore (leggibile, in complemento a due, anche dalla locazione 144 decimale del C/64, cioè $90 esadecimale) la quale riporta l'esito dell'ultima operazione di I/O secondo alcuni significati associati a ciascuno degli 8 bit di ST, come da figura 9.

Si può notare che ai fini del nostro discorso interessano solamente i bit 7, 6, 1 e 0 da impiegare convenientemente per rendere il software in grado di accorgersi se il file è finito (e non deve quindi attendere ulteriori dati in arrivo), se il disk drive è acceso (o meno), oppure ancora se l'operazione I/O (input / output), in generale, ha avuto qualche problema di temporizzazione.

Alcuni problemi possono anche verificarsi in contemporanea; in tal caso la variabile ST (o la locazione 144 = $90 per chi lavora in Assembly) conterrà il valore corrispondente alla somma degli errori stessi. Per esempio un valore di 66 indica che il file da leggere è terminato (64) e, contemporaneamente, si è verificato un eccessivo intervallo di tempo durante la fase di lettura stessa (2); infatti 64 + 2 = 66.

Per l'applicazione di quanto sopra si rimanda alla figura 10, subroutine che verifica la presenza del disk drive sul bus seriale; la variabile di comodo FL viene posta uguale ad uno, in caso di errore "?DEVICE NOT PRESENT"...

LA DIRECTORY DEL DISCHETTO

Il lettore dovrebbe ora essere in grado di aprire un file, gestirlo correttamente ed infine chiuderlo altrettanto bene.

Vi sono, tuttavia, ulteriori informazioni da conoscere con le quali si possono sfruttare particolari caratteristiche messe a disposizione dal D.O.S. V2.6 del disk drive Commodore 1541.

I nomi assegnati ai file di un floppy disk possono essere formati da un massimo di 16 caratteri alfanumerici (lettere e / o numeri) a patto che non contengano determinati caratteri particolari che assumono specifici significati per l'unità 1541.

Il primo di tali caratteri "vietati" è il simbolo del dollaro ($), con il quale viene individuato l'indice del dischetto, cioè la directory.

Su un Commodore 64 è possibile caricare la directory come se fosse un programma Basic (cancellando, però, in tal modo quanto si aveva precedentemente in memoria), digitando il comando...

LOAD "$",8

...ed in seguito impartendo LIST.

Se l'indice non rientra totalmente nel video, la parte superiore scrollerà fuori dalla visuale dello schermo anche se con il tasto Control si potrà rallentare tale scorrimento mentre con il Run/Stop lo si interromperà del tutto.

Caricando, per esempio, la directory del dischetto accluso al fascicolo "Speciale Drive", si noterà quanto segue: all'estrema sinistra della riga superiore si troverà uno zero che rappresenta un numero distintivo per il drive; nelle unità a doppio drive (rarissime ed appartenenti alla vecchia generazione dei computer Commodore) si potrà avere 0 oppure 1, mentre in un 1541 si avrà sempre lo zero.

Accanto a questo, in reverse, viene indicato il nome del dischetto (per un massimo di 16 caratteri), seguito dai due caratteri dell'identificatore (Id.), informazioni fornite al drive al momento della formattazione del dischetto.

Al termine della riga superiore è quindi normalmente posta l'indicazione "2A" la quale indica che il disk drive 1541 utilizza la versione 2A del D.O.S. Commodore.

Dopo la prima linea trovano posto tante righe (ciascuna delle quali fornisce tre elementi di informazione) relative a ciascuno dei file contenuti sul dischetto.

All'estremità sinistra di ciascuna riga si trova la dimensione del file (in blocchi di 254 caratteri): un valore lungo 4 blocchi, per esempio, identifica un file che, nella memoria del computer, occupa quasi un Kilobyte di memoria (254 x 4 = 1016).

La parte centrale di ciascuna linea contiene il nome del file, posto tra le virgolette (aggiunte automaticamente dal D.O.S.), mentre l'estremità destra fornisce un'abbreviazione di tre lettere concernente il tipo di file secondo quanto già visto: PRG, SEQ, REL oppure USR; il DEL, normalmente, non compare dal momento che si tratta di un tipo di file "illegale", nel senso che non risulta disponibile nella directory.

L'ultima riga dell'indice, infine, contiene un'indicazione dei "blocks free" (= blocchi liberi), da 254 caratteri ciascuno, disponibili per l'uso; tale numero varia da 664 (dischetto appena formattato), a 0 (floppy completamente pieno).

L'USO DI INDICI SELETTIVI

Si è visto che con la sintassi LOAD "$",8 (e quindi il LIST) è possibile caricare in memoria, e visualizzare, la directory di un dischetto; vi è però un'ulteriore possibilità, un po' meno conosciuta, per caricare i nomi relativi solo ad alcuni tipi di file.

La sintassi è la seguente:

LOAD "$0: configurazione = tipo del file", 8

Il simbolo del dollaro ($) rappresenta la directory stessa, mentre lo zero che segue subito dopo indica il numero di drive, che può essere uno oppure zero: si è già visto che nel 1541 vale sempre zero in quanto trattasi di un drive singolo.

E' bene distinguere tra questo valore di numero di drive (0 oppure 1) ed il numero di periferica (o di dispositivo, o di device, che è lo stesso) il quale varia generalmente tra 8 e 11. Nei vecchi(ssimi) sistemi Commodore esistevano unità a dischi doppie che, individuate dal numero di periferica 8, mettevano a disposizione il primo o il secondo drive, a seconda del valore (0 oppure 1) che veniva digitato. Esempio:

LOAD "$0",8

...carica la directory del drive 0 con numero di periferica 8.

LOAD "$1",8

...carica la directory del drive 1 con numero di periferica 8.

Tutto ciò, comunque, ha una scusa puramente didattica e non interessa all'utente 1541 il quale potrà impiegare indifferentemente la sintassi LOAD "$0", 8 oppure LOAD "$", 8 per ottenere lo stesso identico risultato.

Ma torniamo alla descrizione del comando: dopo lo zero vi è il segno di doppio punto (:), altro carattere "vietato" nei nomi di file; il disk drive lo interpreta come segno divisore tra un comando (in questo caso il "$" che indica la directory) ed eventuali specifiche del comando.

Nell'esempio, tali specifiche riguardano la selezione di particolari tipi di file nell'ambito dell'intera directory.

Nel D.O.S. V2.6 del 1541 (ed anche in altri) il carattere di asterisco (*) assume il valore di una qualsiasi sequenza di caratteri in sostituzione (parziale o totale) di un nome di file.

Nel caso parziale significa che per indicare un nome si possono digitare solo le prime lettere dello stesso, sostituendo le rimanenti con l'asterisco: "PIP*" indica tutti i nomi che iniziano con la sequenza PIP come, ad esempio, PIPPO, PIPINO, PIPER, PIPA etc.

Nel caso in cui l'asterisco sia il primo carattere del nome del file (caso di sostituzione totale del nome del file), il significato è...

"considera il file formato da una qualsiasi sequenza di caratteri"

...cioè, in pratica, il D.O.S. si riferirà al primo file della directory se il drive è stato appena acceso oppure resettato; in caso contrario terrà per buono l'ultimo nome di file utilizzato.

La configurazione del comando, quindi, può essere formata da un nome di file specifico (e la directory caricata conterrà solo quello) oppure da una configurazione parziale tipo "A*" che indica tutti i file che iniziano con la "A", aventi un nome di lunghezza qualunque.

Se, invece, la lunghezza è nota ma non si è sicuri di ricordare qualche carattere, si può ricorrere all'altro carattere riservato, il punto interrogativo (?).

La sequenza "CI?O", per esempio, identificherà il (oppure i) file che iniziano con le lettere "CI" seguite da un qualsiasi carattere e terminanti con la lettera "O" come "CIAO", "CIBO", "CICO", "CIRO" etc.

Per quanto riguarda il tipo del file, infine, sarà sufficiente indicare la lettera iniziale del tipo di file tra quelli esistenti, tranne la D di DEL chiaramente "vietata".

Ecco, dunque, alcuni esempi pratici d'impiego:

LOAD "$:ABC*", 8

Carica, dalla directory, tutti i nomi di file che iniziano con "ABC".

LOAD "$:ABC*=S", 8

Carica, dalla directory, tutti i nomi dei file di tipo SEQ, che iniziano con "ABC"

LOAD "$:*=P", 8

Carica solamente i nomi dei file di tipo PRG aventi nome qualsiasi di lunghezza qualsiasi.

LOAD "$:?=U", 8

Carica solamente i nomi dei file di tipo USR aventi un nome qualsiasi lungo un solo carattere

LOAD "??ABC*", 8

Carica il primo file della directory che inizia con due caratteri qualsiasi, seguiti da "ABC" e quindi da un'altra sequenza di caratteri qualsiasi; esempio di possibili nomi validi: EFABCD, 33ABCTYS,

UN COMANDO SCONOSCIUTO

Una trattazione completa sul più misterioso tra i comandi del drive 1541: l'utility loader "&"

Nel drive 1541 è nota l'implementazione di una dozzina di comandi dei quali solo la metà sembrano essere di dominio pubblico: tali comandi, da inviare al drive attraverso il canale con indirizzo secondario uguale a 15, sono tutti formati da un carattere che il D.O.S. adopera per attivare le routine atte ad interpretare correttamente i caratteri successivi e, quindi, il comando stesso. Per esempio...

OPEN 1, 8, 15, "I0": CLOSE 1

...inizializza (comando I) il drive numero 0 (il carattere che segue la I).

Ciascuno dei comandi (ad eccezione del comando di backup "D", che però non ha una propria routine esecutiva) svolge una funzione particolare che consente di avere un pieno controllo sul 1541, agendo sia sulla memoria del drive (comandi "M", "U"), che su quella di massa del dischetto (comandi "V", "P", "C", "R", "S", "N") oppure su entrambe (comandi "I", "B", "&").

Proprio l'ultimo comando citato rappresenta, insieme al comando "P" per i file relativi, un comando tanto potente quanto misterioso. Ma mentre ci si è preoccupati di illustrare l'utilizzo dei file relativi (ed annessi comandi) in diversi articoli contenenti i giusti programmi d'impiego dei REL, non si è fatto altrettanto a proposito degli USR, tipo di file impiegato anche dal comando "&" di utility loader.

Tale nome è quello riportato nei manuali Commodore, dal drive 1571 in poi, sebbene esista anche nel buon vecchio 1541. Si tratta, sostanzialmente, di un'evoluzione del comando di Block-Execute il quale può caricare un intero blocco del dischetto in uno dei cinque buffer posti da $300 in poi (a gruppi di 256 byte), per poi eseguirne il codice macchina a partire dal primo byte.

Un file di utility loader, invece, può avere una lunghezza qualsiasi, tenendo conto che al suo interno può contenere una pluralità di sotto-routine (lunghe al massimo 256 byte ciascuna) le quali possono essere caricate a partire da qualsiasi indirizzo R.A.M. del drive. Al termine del caricamento del file di utility loader (generalmente costituito da una sola routine) il codice L.M. della prima di tali routine verrà eseguito a partire dal primo byte caricato: in altre parole l'indirizzo di caricamento sarà anche quello della "SYS" di partenza.

IL FORMATO DEGLI UTILITY LOADER

Si è visto che tutti i file U.L. (Utility Loader) sono del tipo utente, cioè USR, ma ciò non rappresenta nulla di strano, in quanto sono concatenati esattamente come un file programma (PRG) o sequenziale (SEQ): si tratta solo di una convenzione del D.O.S. (o, meglio, dei signori che l'hanno scritto) per dare una prima identificazione al file.

Oltre che essere di tipo USR, infatti, il file deve contenere una (è il caso più comune) o più routine, ciascuna scritta in accordo con il seguente particolare formato (vedi anche figura 11):

I primi due byte dell'U.L. contengono l'indirizzo di caricamento a partire dal quale saranno posti i byte dal quarto (compreso) in poi della routine: nel terzo byte sarà contenuto il numero di byte formanti il codice L.M. vero e proprio (escludendo quindi i due byte dell'indirizzo di caricamento, il byte di lunghezza e quello di checksum). A partire dal quarto byte sarà presente la routine che si vuole far girare nella R.A.M. del 1541, conclusa da un byte di checksum che va calcolato come segue: si sommano i due byte dell'indirizzo di caricamento, quello della lunghezza del codice in linguaggio macchina e tutti i byte L.M. successivi: ogni volta che il totale supera 255 ($FF in esadecimale) viene aggiunto alla checksum un carry (= riporto); il valore risultante dopo aver sommato l'ultimo byte L.M. rappresenta la checksum da "accodare" all'ultimo byte stesso.

Il calcolo della checksum sarà controllato, dal D.O.S. del 1541, da una routine (della quale si riporta il disassemblato commentato) che è stata inclusa nel programma "Utility Loader Maker" il quale, come vedremo in seguito, consentirà di preparare piuttosto facilmente una routine di U.L. da impiegarsi senza modifiche oppure da "appendere" ad altre routines per formare un unico file Utility Loader.

UN ESEMPIO PRATICO

Vista la teoria si può ora passare alla pratica con il seguente problema:

"E' possibile disporre di un comando "&" con il quale sia possibile cambiare a piacere il numero di periferica del drive?"

Nel 1571 esiste un tale comando, con sintassi...

OPEN 1, dv, 15, "U0>33 + CHR$(dn): CLOSE 1

...con il quale si cambia il numero di device da "dv" a "dn". Nel 1541 il D.O.S. non riconosce lo stesso comando che, comunque, può essere creato per mezzo di una routine "&" di utility loader.

Il primo passo è quello di asegnare un nome al nuovo comando: dalle iniziali inglesi di Device Number (numero di periferica), lo si può chiamare "DN". Un programmino l.m. idoneo allo scopo può essere quello riportato in figura 12, ma il lettore potrà apportare le modifiche che desidera.

Il disassemblato presentato non fa altro che prelevare il nuovo numero di device dalla locazione $204 e trasformarlo in "corrente" dopo due necessarie operazioni di OR con i valori di #$20 per il LISTEN e di #$40 per il TALK.

Ma come è possibile che il numero di periferica che abbiamo in mente si trovi proprio nella locazione $204 e non da un'altra parte? Eccone la motivazione:

Il comando di Utility Loader "&", essendo tale, deve essere inviato su un file dati avente 15 come indirizzo secondario, il cosiddetto canale dei comandi.

Nella mappa di memoria del 1541 si legge che esiste un buffer da 512 ($200) a 552 ($228) nel quale vengono depositati tutti i comandi (caratteri, numeri, valori vari etc.) provenienti dal computer; tra questi figurerà anche il nome della nostra routine "&DN" il quale occupa tre locazioni: la "&" andrà in $200, la "D" in $201 e la "N" in $202.

Si potrebbe quindi mettere, come suffisso del nome, il valore (come CHR$) del nuovo device number desiderato ma, in tal caso, esso verrà interpretato come parte integrante del nome che, invece, è costituito da tre soli caratteri: "&DN".

Si risolve la questione ponendo un carattere di separazione (come la virgola, per esempio) tra l'ultima lettera del nome (la "N") ed il CHR$ del nuovo numero di periferica; ecco dunque la sintassi del nostro comando di Utility Loader...

```
-------------------- FIGURA 11 ----------

 Byte della
routine U.L.   Contenuto
------------   ---------
     1         Byte basso dell'indirizzo di caricamento
     2         Byte alto dell'indirizzo di caricamento
     3         Numero di bytes di codice L.M. che seguono, con la
               particolarita' che $00 = 256 bytes
   4 - ??      Bytes del codice eseguibile di linguaggio macchina
  ?? + 1       Byte di checksum di tutti i bytes precedenti

----------------------------------------
```

```
-------------------- FIGURA 12 ----------

.> C000  BB 01                 Indirizzo di caricamento $1BB
.> C002  0E                    Lunghezza del codice in l.m. = 14 bytes = $0E
                               in esadecimale
.> C003  AD 04 02 LDA $0204    Carica il nuovo numero di periferica
.> C006  AA       TAX          Lo parcheggia nel registro X
.> C007  09 20    ORA #$20     Esegue l'OR con #$20 come numero di periferica per
                               la ricezione dei dati (LISTEN)
.> C009  85 77    STA $77      Lo pone nella locazione di LISTEN
.> C00B  8A       TXA          Ricarica il numero di periferica dal registro X
.> C00C  09 40    ORA #$40     Esegue l'OR con #$40 come numero di periferica
                               per l'invio dei dati (TALK)
.> C00E  85 78    STA $78      Lo pone nella locazione di TALK
.> C010  60       RTS          Fine della "&" routine di Utility Loader

----------------------------------------
```

OPEN 1, dv, 15, "&DN," + CHR$(dn): CLOSE 1

...con il quale si potrà svolgere un'operazione analoga a quella già vista nel 1571: il device number cambierà da "dv" (vecchio) a "dn" (nuovo); ma questo solo quando il comando sarà pronto: si era infatti rimasti alla locazione $202 del buffer dei comandi, nella quale era contenuta la "N", ultima lettera del nome.

Con la sintassi suddetta, quindi, la virgola (che andrà in $203) comunicherà al D.O.S. che i caratteri/e successivi non fanno parte del nome ed il valore del CHR$ si troverà in $204, locazione dalla quale il programmino l.m. lo adopererà convenientemente onde trasformarlo in nuovo numero di periferica.

Il programma in linguaggio macchina è stato assemblato nella R.A.M. del Commodore 64 a partire da 49152 ($C000) e, come si nota, manca di un solo particolare dopo il RTS, e cioè del byte di checksum.

Senza preoccuparsi della faccenda, si può comunque registrare su disco (tramite l'abituale monitor di linguaggio macchina), la zona compresa tra $C000 e $C011, chiamandola pure "DN", creando un file "oggetto" pronto per essere trasformato in un eseguibile di tipo USR.

IL PROGRAMMA UTILITY LOADER MAKER

Si è già visto che le routine di tipo "&" sono poco usate a causa, molto probabilmente, della carenza di adeguate informazioni in proposito che ne rendano semplice l'impiego a chi può aver bisogno di potenza e flessibilità.

Una notevole seccatura è rappresentata, anche per chi conosce gli U.L., dal calcolo (possibilmente esatto!) della checksum da porre al termine del codice di linguaggio macchina e dalla necessità di trasformare in USR la routine.

Se, per quest'ultima, è sufficiente un SAVE "nome,U", 8, per il caso checksum è indubbia la necessità di disporre di un'apposita routine che svolga un'operazione tanto semplice quanto induttiva in irritanti errori che il comando "&" può generare.

Se la checksum non è corretta, infatti, verrà restituito un errore numero 50 di "Record not present", così come se il byte di lunghezza del codice L.M. contiene un valore minore di quello reale (un codice di L.M. verrà assunto come chechsum, ovviamente sbagliata!) mentre se si riporta un valore di lunghezza maggiore di quello effettivo si avrà un errore 51 di "Overflow in record".

Il pericolo della checksum, comunque, potrà essere scongiurato da un programma presente sul dischetto allegato allo speciale, chiamato Utility Loader Maker, il quale provvederà a tramutare il file PRG chiamato "DN" senza checksum, in un file USR chiamato "&DN" con la checksum corretta, pronto per essere eseguito da un comando di OPEN (come sopra riportato) o da un comando di PRINT#.

Verrà cioè inizialmente chiesto il nome del file di tipo programma da convertire e verrà svolto tutto il lavoro necessario, completato da una routine di gestione errori che avvertirà di eventuali problemi incontrati durante l'operazione che offrirà, alla fine, due file sul dischetto: quello originale, chiamato "DN", e quello eseguibile come comando "&" di U.L. chiamato "&DN". I due file sono comunque presenti sul dischetto allegato.

L'UTILIZZO DEI COMANDI "&" DI U.L.

Una volta creato il nostro file U.L. è possibile farlo girare nella R.A.M. del disk drive 1541 con un semplice comando inviato sul canale secondario 15 come comando "&": ecco due sintassi basic equivalenti per cambiare il numero di periferica da 8 a 9 con il nuovo comando appena realizzato:

OPEN 1, 8, 15, "&DN," + CHR$(9): CLOSE 1
OPEN 1, 8, 15: PRINT# 1, "&DN,"; CHR$(9): CLOSE 1

E' questo il cosiddetto metodo software che richiede, obbligatoriamente, la presenza della "&" come primo carattere del nome, presenza non necessaria impiegando invece un altro sistema di attivazione della routine U.L. detto, in contrapposizione al primo, metodo hardware.

Tale modalità è resa possibile dalla presenza nella R.O.M. del D.O.S. del 1541 di una routine, detta di controllo dell'autostart, che inizia (o meglio iniziava) dalla locazione $E780.

Sebbene non presente nella nuova release del D.O.S. 1541 e del 1571, di tale routine si riporta comunque, per comple-

tezza, il disassemblato (ovviamente commentato, riquadro 2) dal quale si può dedurre la modalità operativa del metodo hardware.

Tenendo a massa (pin numero 2 del bus seriale) le linee CLK cioè CLOCK (pin numero 4) e DATA (pin numero 5), il 1541 attende il ritorno delle linee allo stato logico 0 (+5 Volt) per caricare ed eseguire il primo file presente sul dischetto inserito nel disk drive.

UN POTENTE COMANDO

Da queste righe dovrebbe essere emersa l'importanza del comando descritto che, in quanto "costruito" dall'utente, possiede una potenza intrinseca notevole.

L'invito è ovviamente rivolto alla considerazione del programma "&DN" come di un puro esempio (sebbene alquanto utile ed immediato) di quanto possa essere realizzato con il non più misterioso comando "&" di U.L. grazie anche alle note esposte nell'articolo e nel programma U.L. Maker (riquadro 1) il quale solleva persino dal compito più noioso inerente la creazione di un file USR di Utility Loader.

Solo nel campo delle protezioni, ad esempio, si potrebbero avere "&" routine piuttosto "cattive" che formattano dischi, disallineano testine o, molto più semplicemente, cancellano tutte le tracce tranne la diciottesima...

```
------------- REM RIQUADRO 1 ----------

PROGRAMMA "UTILITY LOADER MAKER"
---------------------------------

DISASSEMBLATO COMMENTATO:


Bytes basic della SYS di partenza

.> 0800 00 13 08 C4 07 9E 32 30: ...D..20
.> 0808 36 39 14 14 14 14 54 45: 69....TE
.> 0810 4D 53 00 00 00          : MS...


.> 0812  00        BRK        ;
.> 0813  00        BRK        ;
.> 0814  00        BRK        ;
.> 0815  A2 00     LDX #$00   ; Colore nero...
.> 0817  8E 20 D0 STX $D020   ; ...per la cornice...
.> 081A  8E 21 D0 STX $D021   ; ... e per il bordo
.> 081D  BD 3F 09 LDA $093F,X; Scrive l'intestazione
.> 0820  F0 06     BEQ $0828  ; Esce se il carattere e' nullo
.> 0822  20 D2 FF JSR $FFD2   ; Invia allo schermo il carattere
.> 0825  E8        INX        ; Continua...
.> 0826  D0 F5     BNE $081D  ; ... sino alla fine
.> 0828  A0 A7     LDY #$A7   ; Riempie la zona
.> 082A  A9 A0     LDA #$A0   ; da $2A7 a $2FF
.> 082C  99 00 02 STA $0200,Y; con lo spazio shiftato
.> 082F  C8        INY        ; per posizionarvi
.> 0830  D0 FA     BNE $082C  ; il nome del file
.> 0832  A9 26     LDA #$26   ; Carattere "&"...
.> 0834  8D A7 02 STA $02A7   ; ...prima del nome
.> 0837  20 CF FF JSR $FFCF   ; Input del nome del file da convertire
.> 083A  C9 0D     CMP #$0D   ; E' un Return?
.> 083C  F0 06     BEQ $0844  ; Esci se affermativo
.> 083E  99 A8 02 STA $02A8,Y; Altrimenti poni il nome da $2A8 in poi
.> 0841  C8        INY        ; Prosegue con
.> 0842  D0 F3     BNE $0837  ; il prossimo carattere
.> 0844  98        TYA        ; Numero di caratteri del nome
.> 0845  A2 A8     LDX #$A8   ; Byte basso di indirizzo del nome $2A8
.> 0847  A0 02     LDY #$02   ; Byte alto di indirizzo del nome $2A8
.> 0849  20 BD FF JSR $FFBD   ; Assegna lunghezza ed indirizzo del nome del file
.> 084C  A9 00     LDA #$00   ; File numero 0
.> 084E  A2 08     LDX #$08   ; Periferica numero 8 - disk drive
.> 0850  A8        TAY        ; Indirizzo secondario 0 per il LOAD
.> 0851  20 BA FF JSR $FFBA   ; Assegna i parametri del file logico
.> 0854  AA        TAX        ; Byte basso dell'indirizzo di caricamento $C000
.> 0855  A0 C0     LDY #$C0   ; Byte alto dell'indirizzo di caricamento $C000
.> 0857  86 FB     STX $FB    ; Conserva in pagina
.> 0859  84 FC     STY $FC    ; zero l'indirizzo di caricamento
.> 085B  20 D5 FF JSR $FFD5   ; LOAD del file da convertire
.> 085E  CA        DEX        ; Trova il valore
.> 085F  E0 FF     CPX #$FF   ; dell'ultimo byte
.> 0861  D0 01     BNE $0864  ; occupato dal file
.> 0863  88        DEY        ; e lo conserva
.> 0864  86 FD     STX $FD    ; in due locazioni
.> 0866  84 FE     STY $FE    ; della pagina zero
.> 0868  20 B7 FF JSR $FFB7   ; Legge lo Status
.> 086B  C9 40     CMP #$40   ; E' uguale a 64 (fine del file)?
.> 086D  D0 79     BNE $08E8  ; No, errore
.> 086F  A0 00     LDY #$00   ; Si, azzera la checksum
.> 0871  84 FF     STY $FF    ; posta in 255 ($FF)
.> 0873  18        CLC        ; Carry a zero
.> 0874  B1 FB     LDA ($FB),Y; Carica un byte del programma
.> 0876  65 FF     ADC $FF    ; Lo somma alla checksùm corrente
.> 0878  69 00     ADC #$00   ; Somma l'eventuale carry generatosi
.> 087A  85 FF     STA $FF    ; Lo pone in $FF come nuova checksum
.> 087C  A5 FB     LDA $FB    ; Confronta il byte basso dell'indirizzo corrente
.> 087E  C5 FD     CMP $FD    ; e di quello di fine caricamento
.> 0880  D0 06     BNE $0888  ; Prosegui se sono differenti
```

```
.> 0882  A5 FC     LDA $FC    ; Confronta il byte alto dell'indirizzo corrente
.> 0884  C5 FE     CMP $FE    ; e di quello di caricamento
.> 0886  F0 08     BEQ $0890  ; Esci se sono uguali
.> 0888  E6 FB     INC $FB    ; Incrementa il byte basso
.> 088A  D0 E7     BNE $0873  ; Prosegui se non vi e' overflow
.> 088C  E6 FC     INC $FC    ; Incrementa il byte alto
.> 088E  D0 E3     BNE $0873  ; Prosegui con il prossimo byte
.> 0890  A4 B7     LDY $B7    ; Numero di caratteri del nome del file
.> 0892  A2 03     LDX #$03   ; Quattro caratteri (3 + lo zero) da aggiungere in
coda al nome del file
.> 0894  BD 81 09 LDA $0981,X; Trasferisce il suffisso
.> 0897  99 A8 02 STA $02A8,Y; ",U,W" dopo il nome
.> 089A  C8        INY        ; del file e prosegue
.> 089B  CA        DEX        ; per tutti i
.> 089C  10 F6     BPL $0894  ; quattro caratteri
.> 089E  C8        INY        ; Incrementa la lunghezza del nome del file,
considerando anche il carattere "&" aggiunto prima del nome del file
.> 089F  84 B7     STY $B7    ; Nuovo numero di caratteri del nome del file
.> 08A1  C6 BB     DEC $BB    ; Il nome del file inizia un carattere prima (dalla
"&")
.> 08A3  A9 02     LDA #$02   ; Numero di file logico e di indirizzo secondario
.> 08A5  85 B8     STA $B8    ; File logico
.> 08A7  85 B9     STA $B9    ; Indirizzo secondario
.> 08A9  20 C0 FF JSR $FFC0  ; OPEN 2,8,2,"&nomefile,U,W"
.> 08AC  A2 02     LDX #$02   ; Canale numero 2...
.> 08AE  20 C9 FF JSR $FFC9  ; ...aperto in uscita
.> 08B1  AD 00 C0 LDA $C000  ; Primo byte
.> 08B4  20 D2 FF JSR $FFD2  ; Registra sul file USR
.> 08B7  AD 01 C0 LDA $C001  ; Secondo byte
.> 08BA  20 D2 FF JSR $FFD2  ; Registra sul file USR
.> 08BD  AD 02 C0 LDA $C002  ; Terzo byte
.> 08C0  20 D2 FF JSR $FFD2  ; Registra sul file USR
.> 08C3  AE 02 C0 LDX $C002  ; Numero di bytes del codice in linguaggio macchina
.> 08C6  A0 00     LDY #$00   ; Indice uguale a zero
.> 08C8  B9 03 C0 LDA $C003,Y; Prende un byte del codice L.M.
.> 08CB  20 D2 FF JSR $FFD2  ; Lo registra sul file USR
.> 08CE  C8        INY        ; Incrementa l'indice
.> 08CF  CA        DEX        ; Decrementa il numero di bytes del codice
.> 08D0  D0 F6     BNE $08C8  ; Prosegui se diverso da zero
.> 08D2  A5 FF     LDA $FF    ; Valore della checksum
.> 08D4  20 D2 FF JSR $FFD2  ; Registra sul file USR
.> 08D7  A9 02     LDA #$02   ; File logico 2
.> 08D9  20 C3 FF JSR $FFC3  ; CLOSE del file logico numero 2
.> 08DC  20 E7 FF JSR $FFE7  ; Chiude tutti i files e ripristina le periferiche
I/O di default
.> 08DF  20 B7 FF JSR $FFB7  ; Legge lo Status
.> 08E2  F0 22     BEQ $0906  ; Prosegue se uguale a zero
.> 08E4  C9 40     CMP #$40   ; E' uguale a 64 (fine del file)?
.> 08E6  F0 1E     BEQ $0906  ; Si, prosegui

Routine del campanellino di segnalazione d'errore

.> 08E8  A9 15     LDA #$15   ; Valore massimo del volume del S.I.D.
.> 08EA  A2 09     LDX #$09   ; Attack/decay
.> 08EC  A0 00     LDY #$00   ; Sustain/release
.> 08EE  8D 18 D4 STA $D418  ; Volume del S.I.D.
.> 08F1  8E 05 D4 STX $D405  ; Attack/decay per la voce 1
.> 08F4  8C 06 D4 STY $D406  ; Sustain/release per la voce 1
.> 08F7  A9 30     LDA #$30   ; Byte high della frequenza
.> 08F9  A2 20     LDX #$20   ; Seleziona il dente di sega
.> 08FB  A0 21     LDY #$21   ; Valore d'inizializzazione del suono
.> 08FD  8D 01 D4 STA $D401  ; Voce numero 1
.> 0900  8E 04 D4 STX $D404  ; Forma d'onda
.> 0903  8C 04 D4 STY $D404  ; Forma d'onda
.> 0906  A0 00     LDY #$00   ; Indice a zero
.> 0908  B9 85 09 LDA $0985,Y; Prende un carattere della scritta "Stato disco:"
.> 090B  F0 06     BEQ $0913  ; Esce se nullo
.> 090D  20 D2 FF JSR $FFD2  ; Altrimenti lo invia allo schermo
.> 0910  C8        INY        ; Incrementa l'indice
.> 0911  D0 F5     BNE $0908  ; Prosegue con il prossimo carattere
.> 0913  A9 08     LDA #$08   ; Periferica numero 8 (disk drive)
.> 0915  20 B4 FF JSR $FFB4  ; Invia TALK sul bus seriale
.> 0918  A9 FF     LDA #$FF   ; Indirizzo secondario 15 ($0F) e codice di
apertura ($F0): $0F OR $F0 - $FF
.> 091A  20 96 FF JSR $FF96  ; Invia l'indirizzo secondario con TKSA
.> 091D  20 A5 FF JSR $FFA5  ; Riceve un byte dal bus seriale (carattere dello
stato disco)
.> 0920  20 D2 FF JSR $FFD2  ; Lo invia allo schermo
.> 0923  C9 0D     CMP #$0D   ; E' un Return?
.> 0925  D0 F6     BNE $091D  ; No, prosegui con il prossimo carattere
.> 0927  20 AB FF JSR $FFAB  ; Invia UNTALK sul bus seriale
.> 092A  A0 00     LDY #$00   ; Indice a zero
.> 092C  B9 97 09 LDA $0997,Y; Prende un carattere della scritta "Premere un
tasto"
.> 092F  F0 06     BEQ $0937  ; Esce se nullo
.> 0931  20 D2 FF JSR $FFD2  ; Altrimenti lo invia allo schermo
.> 0934  C8        INY        ; Incrementa l'indice
.> 0935  D0 F5     BNE $092C  ; Prosegui con il prossimo carattere
.> 0937  20 E4 FF JSR $FFE4  ; E' stato premuto un tasto?
.> 093A  F0 FB     BEQ $0937  ; No, aspetta
.> 093C  4C 15 08 JMP $0815  ; Causa nuovamente il RUN del programma
```

Scritte d'intestazione:

```
.>0093F 93 08 0E 20 20 20 20 20: ...
.>00947 20 98 2D 3D 2A 20 05 D5:  .-=* .U
.>0094F D4 C9 CC C9 D4 D9 20 CC: TILITY L
.>00957 CF C1 C4 C5 D2 20 CD C1: OADER MA
.>0095F CB C5 D2 20 98 2A 3D 2D: KER .*=-
.>00967 0D 0D 0D 98 C6 49 4C 45: ....FILE
.>0096F 20 44 41 20 43 4F 4E 56:  DA CONV
.>00977 45 52 54 49 52 45 3F 20: ERTIRE?
.>0097F 05 00                   : ..
```

Suffisso del tipo di file "User" in modo "Write" per la scrittura dell'"&-File utility loader":

```
.>00981 57 2C 55 2C             : W,U,
```

Scritta di informazione dello stato del disco:

```
.>00985 0D 0D 98 D3 54 41 54 4F: ...STATO
.>0098D 20 44 49 53 43 4F 3A 20:  DISCO:
.>00995 05 00                   : ..
```

Scritta di conclusione del programma:

```
.>00997 0D 0D 98 D0 52 45 4D 45: ...PREME
.>0099F 52 45 20 55 4E 20 54 41: RE UN TA
.>009A7 53 54 4F 2E 00          : STO..
```

------------- REM RIQUADRO 2 ----------

ROUTINE R.O.M. DEL D.O.S. DI CONTROLLO PER L'AUTO-START

Nota: non presente nelle nuove R.O.M. del disk drive 1541

DISASSEMBLATO COMMENTATO:

```
.> E77F  60       RTS        ; Fine della routine
.> E780  AD 00 18 LDA $1800  ; Legge la porta IEEE
.> E783  AA       TAX        ; Ne parcheggia il valore nel registro X
.> E784  29 04    AND #$04   ; Isola il bit di Clock In
.> E786  F0 F7    BEQ $E77F  ; Esce se non e' ad uno
.> E788  8A       TXA        ; Riprende dal registro X la lettura della porta
IEEE
.> E789  29 01    AND #$01   ; Isola il bit di Data In
.> E78B  F0 F2    BEQ $E77F  ; Esce se non e' ad uno
.> E78D  58       CLI        ; Riabilita l'interrupt
.> E78E  AD 00 18 LDA $1800  ; Legge nuovamente la porta IEEE
.> E791  29 05    AND #$05   ; Testa contemporaneamente i bits Clock In e Data
In
.> E793  F0 F9    BNE $E78E  ; Attende entrambi a zero
.> E795  EE 78 02 INC $0278  ; Nome del file
.> E798  EE 74 02 INC $0274  ; Caratteri nella linea di input
.> E79B  A9 2A    LDA #$2A   ; Carattere "*" come nome del file
.> E79D  8D 00 02 STA $0200  ; Lo scrive nel buffer
.> E7A0  4C A8 E7 JMP $E7A8  ; Prosegue come per il comando "&"
```

------------- REM RIQUADRO 3 ----------

ROUTINE R.O.M. DEL D.O.S. DI CALCOLO DELLA CHECKSUM PER I FILE USR DEL COMANDO "&" DI UTILITY LOADER

DISASSEMBLATO COMMENTATO:

```
.> E84B  A8       CLC        ; Azzera il carry
.> E84C  65 87    ADC $87    ; Somma la vecchia checksum al valore contenuto
nell'accumulatore
.> E84E  69 00    ADC #$00   ; Somma l'eventuale carry generatosi
.> E850  85 87    STA $87    ; Lo pone come nuovo valore di checksum
.> E852  60       RTS        ; Fine della routine
```

GLI ERRORI "PROTETTIVI"

Come mai i dischi dei programmi in commercio sono pieni di errori?

"Errare humanum est" ricordava Seneca nelle sua celebre frase tratta dalle "Declamazioni; sebbene San Bernardo vi fece un "append" del suo invito a non essere diabolici (...humanum tamen sed diabolicum!) sembra che il consiglio non abbia avuto un riscontro pratico, perlomeno dal punto di vista informatico, dal momento che a tutti sarà capitato di osservare il furioso lampeggiare del Led del drive durante l'uso di un qualsiasi programma commerciale, presunto originale.

Tale sfolgorio obbligatorio, al quale si è costretti ad assistere, è stato dettato dall'abitudine leggermente screanzata assunta da alcuni figuri più o meno loschi i quali erano soliti (ri)mettere in vendita i programmi copiati da altre software-house le quali, contemporaneamente, stavano "cercando" di vendere la versione originale ad un prezzo (indovinate un po'?) maggiore, visto che loro, sfortunatamente, il software l'hanno dovuto creare davvero anzichè trascriverlo da un altro dischetto.

In attesa di una legge che obblighi i programmatori e le software-house a lavorare completamente GRATIS le ditte produttrici si sono affidate ad espedienti di ogni tipo per la disperata protezione dei lavori dei loro bravissimi, e costosissimi, programmatori.

Hanno quindi esplorato il C/64 alla ricerca di puntatori da alterare, ipermegautostart disabilitanti l'intero sistema elaborativo ed altre diavolerie più o meno valide contro le quali si poteva sempre impiegare il "Galacopier V999.99" il cui numero di release si incrementava ad ogni uscita di una nuova protezione, creando, di fatto, una gara senza fine.

La competizione, tuttora in corso, si è decisamente trasferita su altri ben più veloci circuiti, mentre una costante è rimasta nel rapidissimo progredire dei sistemi anticopia: gli errori sul dischetto.

IL MONDO COPIATORI

Sostanzialmente un copiatore ha il compito, apparentemente semplice, di ricreare su un dischetto - copia un ben preciso stato magnetico rilevato dal floppy disk originale.

Il compito viene generalmente svolto leggendo le informazioni da quest'ultimo supporto e depositandole temporaneamente nella R.A.M. del computer dalla quale verranno, in seguito, ripresi per il finale trasferimento sulla copia.

Le caratteristiche del D.O.S. del 1541 hanno, però, suggerito una semplice procedura per impedire la lettura di alcuni settori del disco da parte della R.O.M. del 1541, la quale esegue prima durante e dopo le operazioni di I/O, una spaventosa quantità di controlli; in caso di fallimento di uno solo di questi, il drive genererà inesorabilmente un messaggio di errore, impedendo la prosecuzione delle operazioni.

Solitamente l'operazione particolare che viene impedita è proprio quella della lettura dal disco originale, mediante la creazione intenzionale, sullo stesso, di alcuni errori, la cui presenza andrà poi verificata da apposite routine, come si specificherà tra breve.

I vecchi copiatori, quindi, erano costretti a trasferire la sola parte "leggibile" del dischetto. Quando il programma veniva mandato in esecuzione, e falliva il controllo a causa della presenza degli errori, si otteneva l'impiantamento o l'autocancellazione del software che impediva l'accesso al programma vero e proprio.

Oggigiorno i programmi vengono scritti su disco in particolarissimi formati che risultano alquanto impenetrabili da chiunque non abbia una più che buona conoscenza del D.O.S. V2.6 e magari di qualche altro sistema operativo per dischi (tipo la gestione drive del programma di protezione "Libra" per l'ambiente MS-DOS, dalla quale si può imparare davvero parecchio!).

"Esteticamente" è solo rilevabile un gruppo di settori completamente pieni di errori i quali, in realtà, nascondono i veri dati (scritti, comunque, in maniera codificata) in traccie "randomatiche" la cui distanza e distribuzione è data dai codici di una particolare password (la quale può anche essere "incorporata" e non necessariamente chiesta all'utente) che andrà anche a decodificare i dati letti: diabolicamente machiavellico!

Esamineremo ora, in dettaglio, la tradizionale protezione basata sugli errori, sebbene oggi non sia più in vigore nelle softhouse di un certo livello.

PROGRAMMI ED ERRORI

La protezione dei programmi mediante la sola creazione di errori su disco oggigiorno non è più affidabile.

I copiatori più diffusi prevedono anche alcune routine che, girando nella R.A.M. del 1541, "by-passano" la tradizionale R.O.M. di lettura del D.O.S., evitando, di conseguenza, tutte le generazioni di errori non desiderate.

La creazione di tali routine di lettura alternative, almeno in teoria, è sostanzialmente molto semplice: viene ricopiata in R.A.M. pari pari, la versione R.O.M. originale e, quindi, "elaborata" togliendo tutte le istruzioni di controllo della presenza di eventuali errori chè, in caso di riscontro positivo, abortirebbero la procedura di lettura.

Il risultato è una routine scarnissima che svolge una sola operazione: la lettura di un qualsiasi settore.

Mentre, a questo punto, il D.O.S. V2.6 controlla che il primo valore del blocco dati sia uguale a 7 per non generare il messaggio di errore "22, READ ERROR etc.", (indicante un blocco dati non presente), la routine "fantasma" esegue regolarmente la lettura del settore senza curarsi della validità, o meno, dei valori "di contorno".

Ed ecco che, di riflesso, si intuisce anche come si crea un falso errore su dischetto: l'errore "23, READ ERROR etc." (che sino a pochissimi anni fa sembrava invalicabile!), per esempio, viene generato se il valore della somma di controllo (checksum)

del blocco dati è differente dal valore corretto calcolato al momento della scrittura del settore stesso: basta ricorrere alla routine R.O.M. di scrittura copiata in R.A.M. ed opportunamente modificata, in modo che il calcolo dell'esatta checksum non venga considerato e si riporti sul dischetto un qualsiasi altro valore il quale, in fase di lettura, darà luogo ad un non più misterioso errore numero 23.

La creazione di un errore su disco, quindi, richiede sostanzialmente un'adatta routine che deve obbligatoriamente girare nella R.A.M. del drive in quanto l'interazione con il D.O.S. è troppo diretta per essere compiuta dall'esterno.

Tutti gli accessi al dischetto, infatti, vengono trasformati in alcuni parametri depositati in apposite locazioni dalle quali la routine di interrupt del 1541 provvederà a prelevarli per sapere esattamente se deve leggere o scrivere e dove farlo nell'ambito dell'intera superficie magnetica del floppy disk.

La ragione è semplicemente dettata dalle delicatissime temporizzazioni che regolano le funzioni I/O del disk drive in quanto una semplice POKE in l.m. deve trasformarsi nell'univoca modifica (nel caso del Write) di una ben precisa zona del supporto: il singolo byte registrato.

ERRORI: QUALI SONO?

Sono qui riportati due soli esempi di errori per rendere noto che un errore non necessariamente è tale; sebbene velocissimo, un computer è decisamente stupido: basta spostargli anche una sola virgola per "convincere" il software di gestione (nella fattispecie, il D.O.S. V2.6) che vi è un "errore".

Come si sarà notato (delusi?), la creazione di un errore risulta essere alquanto banale sebbene la faccenda non abbia un'applicabilità universale: non tutti gli errori possono infatti essere riprodotti artificialmente su un dischetto (Esempio: il Write Protect On non è ricreabile in lettura) ma quei pochi che possono esserlo, sono stati (ed in qualche caso lo sono tuttora) sufficienti a proteggere i floppy disk dalla duplicazione abusiva.

Sebbene esistono anche alcuni errori di protezione in scrittura, il discorso verterà essenzialmente su quelli in lettura, impiegati in tutte le generazioni di protezioni con errori su disco.

Per esaurire il discorso dei Write Error basterà ricordare la modifica del valore 65 ($41) presente nel byte 2 della traccia 18 settore 0, con un qualsiasi valore differente il quale, nel caso di tentativi di scrittura sul floppy così trattato, genererà l'errore numero 73 di D.O.S. Mismatch ossia non compatibile in Write mode; è però aggirabile con una routine ad accesso diretto che riporti il byte colpevole al valore originale.

E' ancora più semplice la risoluzione del problema dovuto all'errore 26 (Write Protect On) che consente la scrittura con la semplice rimozione della linguetta protettiva.

GLI ERRORI "SERI"

Ed eccoci ai celeberrimi READ ERROR che tante pene hanno fatto patire alle testine dei 1541 dell'intero globo terraqueo.

Il gruppo degli errori simulabili su floppy comprende quattro tipi principali, ai quali se ne sono in seguito aggiunti altri due per un totale di sei "falsità magnetiche" che si andranno or ora ad esplicare.

READ ERROR numero 20

Apparentemente è uno dei più micidiali: il controller del disco non è in grado di localizzare l'intestazione del blocco e quindi non lo legge neppure.

Analizzando COME fa il 1541 a considerare "trovata" l'intestazione di un settore, si può efficacemente scoprire in quale modo fargli credere presente un inesistente errore numero 20.

Il D.O.S. legge l'intestazione di un qualsiasi blocco con la routine presente da $F510 nella R.O.M. del disk drive e confronta i primi otto valori letti con gli otto presenti in pagina zero (ci riferiamo alla mappa della memoria del 1541 e non del computer) da $24 a $2B compresi: qualunque differenza decrementerà un contatore del numero di tentativi (il registro X inizialmente è settato a 90) il quale, raggiunto il valore nullo (tentativi esauriti...) genererà l'errore 20.

Una routine di creazione di tale errore, quindi, di solito legge il settore da alterare, ne modifica gli ormai famosi otto valori, e lo riscrive con una routine su R.A.M. che non controlla la validità dei valori d'intestazione.

READ ERROR numero 21

Questo errore indica che il controller non è in grado di rilevare la presenza di un carattere di sincronizzazione valido.

Dal punto di vista pratico, la realizzazione "artificiale" del presente "READ ERROR" è la più difficoltosa, proprio per il rischio di andare a sovrascrivere zone da preservare a causa della precisione di posizionamento richiesta alla testina quando modifica i corretti caratteri di sincronismo che, normalmente, hanno la configurazione binaria $01010101 cioè $55 esadecimale.

Il n. 21, infatti, è un errore che prescinde da alcuni valori di controllo registrati insieme ad ogni settore (checksum, Id, etc.) ma viene generato a monte della lettura, quando la testina aspetta che una serie di zero ed uno gli passi davanti, per rendersi conto se l'allineamento con il settore da leggere è corretto oppure no.

Come tale non può essere creato in seguito ad una lettura del blocco, ma necessita di un allineamento con l'header dello stesso, una lettura dei valori da non modificare, un posizionamento sul primo dei valori da alterare (il primo dei SYNC), il cambiamento istantaneo del Port Controller da input (Read) ad output (Write) e l'immediato invio alla testina di scrittura di una serie di SYNC illegali, misti ad alcuni validi (altrimenti si rischia di non trovare più il settore per davvero!), in modo che una normale routine R.O.M. non si senta sufficientemente "sicura" di essersi correttamente posizionata sul settore da leggere; il risultato, ovviamente, sarà quindi un "21, READ ERROR, tt, ss".

READ ERROR numero 22

Si tratta del tipico errore da "falso allarme": "Blocco dati non presente" è l'ufficiale descrizione del "22, R. E." il quale, in realtà, è ricreabile con una facilità estrema.

Il D.O.S. V2.6, subito dopo il secondo SYNC che separa l'intestazione dal blocco dati vero e proprio, aspetta di leggere un valore di $07 che rappresenta la costante di inizio dati del blocco.

Letto il 7, quindi, il controller sa che dal successivo dato (compreso!) in avanti, arriveranno i 256 byte del settore scritto sul dischetto.

L'errore numero 22 è intimamente legato a questa costante uguale a 7. Se, infatti, il D.O.S. non la trova al posto giusto (e legge, per esempio, un valore di otto oppure un qualsiasi altro differente dal sette "originale"), verrà generato il READ ERROR di blocco dati non presente, anche se tutti i 256 byte successivi sono correttissimi.

Appurato che il trucco è legato alla modifica della costante 7 (normalmente contenuta in pagina zero all'indirizzo $47), si sarà compreso come generare l'errore numero 22: una lettura (anche con la routine R.O.M.), seguita da un'alterazione della locazione $47 e da una riscrittura del settore sortirà il voluto effetto.

READ ERROR numero 23

Altro errore dell'ex serie "gli impossibili", la cui banalità di creazione è paragonabile a quella del precedente.

Il byte di contorno da "toccare" viene rivelato dalla descrizione dell'errore stesso: "Errore di checksum nel blocco dati".

Già si dovrebbe sapere che per ogni settore vi sono due somme di controllo: la prima per l'intestazione, e la seconda per il blocco dei dati; orbene il "nostro uomo" è la seconda checksum per modificare la quale basterà agire come per l'errore numero 22.

La lettura del settore fornirà la checksum corretta nella locazione $3A dalla quale un programma di creazione errori sofisticato, ad esempio, potrà leggerla e proporla all'utente per farsi quindi dare la nuova (falsa) checksum desiderata che andrà a sovrascrivere la locazione $3A.

A questo punto basterà impiegare, per la riscrittura del blocco, una routine in R.A.M. del tutto identica a quella su R.O.M. con l'eccezione della chiamata alla routine di calcolo dei valori di "contorno" corretti, tra cui la "Chk 2" presente in $3A la quale deve, invece, rimanere "illegale".

READ ERROR numero 27

Il presente è l'errore gemello del precedente: "Errore di checksum nell'intestazione".

Tuttavia la sua generazione è alquanto complessa rispetto al precedente n. 23, dal momento che ci si trova ad operare modifiche nei valori dell'intestazione che richiedono particolari temporizzazioni molto più restrittive rispetto alla zona dei dati del settore: leggendo un byte non è possibile eseguire troppe istruzioni in L.M. in quanto il numero dei cicli di clock necessario per eseguirle può generare incompatibilità con le impostazioni dei timer interni.

Ovviamente la faccenda non è, però impossibile, tant'è vero che si fa: leggendo l'intestazione si provvede a modificare la "Chk 1" mentre si prosegue la lettura (inutile...) di altri dati, necessaria per far quadrare i conti come numero di cicli, rispetto alla routine R.O.M.

Nella locazione $1A si troverà il corretto valore di checksum subito dopo la lettura attraverso la R.O.M. originale (oppure equivalenti routine R.A.M.) e quindi si potrà agire come per la "Chk 2" a proposito dell'errore precedente.

Decisa la checksum errata da scrivere, si potrà eseguire la routine R.O.M. di scrittura posta su R.A.M. e "mutilata" dei controlli di validità intestazioni e simili.

Naturalmente si dovrà prestare attenzione ad eventuali trasferimenti di valori che avvengono nelle routine di calcolo e verifica delle somme di controllo, in maniera da non far "saltare" la delicatissima procedura di creazione di un errore come il 27.

DISK ID MISMATCH numero 29

Anche se può sembrare strano, tale errore ("Errore nell'Id. del dischetto") è intimamente connesso al precedente: entrambi verificano dei valori presenti nell'intestazione ed entrambi vengono verificati dalla stessa routine R.O.M posta a partire da $F3B1.

Ne consegue che le modalità di creazione dei due errori sono pressochè identiche; ovviamente nel caso del 29 la modifica riguarderà uno oppure entrambi i valori dell'Id. che, per dar luogo all'errore, dovranno essere settati a valori differenti dagli originali (i codici A.S.C.I.I. dei due caratteri dell'Id..

Anche in questo caso rimangono validi tutti i problemi dovuti alla delicata modifica dell'intestazione che deve avvenire con routine praticamente identiche a quelle originali della R.O.M. con le minime modifiche necessarie alla generazione dell'errore desiderato.

E PER RIMETTERE TUTTO A POSTO?

Disfare è indubbiamente più facile che creare qualcosa, ed infatti l'eliminazione degli errori è un altro dei miti infranti che ci si accingerà a descrivere.

Solitamente la necessità principale legata alla protezione con errori su disco riguarda la creazione degli stessi e non certo la loro rimozione.

Vuoi per didattica, vuoi per informazione, vuoi per altri motivi leggermente più loschi, è però nato il desiderio di risanare (se possibile) quei settori contenenti alcuni errori, in modo da renderli nuovamente accessibili al D.O.S. V2.6.

Tutto ciò è chiaramente possibile se l'errore è di tipo software, magari creato artificialmente; se con un punteruolo si incide un dischetto, chiaramente, l'errore creato in tal modo non sarà assolutamente rimuovibile con una semplice routine "Elimina Errori".

Normalmente, però, tutti gli errori presenti su un dischetto sono del tipo software e quindi rimuovibili da parte di una specifica routine.

Tale routine ha una struttura sostanzialmente identica a quella delle routine di creazione degli errori; la funzione svolta è quella di leggere nella R.A.M. del drive il settore danneggiato, dopo aver ricavato, da una traccia non danneggiata, i valori corretti di Id., sincronismi di inizio e fine, etc.

La routine di lettura che svolge questo compito è anch'essa del tutto simile alla sua collega su R.O.M. con l'ovvia esclusione dei controlli di errori vari il cui riscontro porterebbe immancabilmente all'interruzione della procedura.

Una volta acquisito il blocco in memoria, il più è fatto; si ha a disposizione un normalissimo settore che si può riscrivere con le routine standard.

Per universalizzare la procedura, però, anche la routine di scrittura viene riscritta su R.A.M. in modo da riparare anche il "terribile" errore numero 21 con tutto il suo codazzo di sincronismi alterati.

Una routine che svolge questo delicato compito, sebbene non verrà descritta con disassemblato, è stata comunque inserita nel dischetto (allegato al presente "Speciale") con il nome di "Elimina Errori"; sebbene il programma sia già pronto per l'uso, sarà comunque possibile l'estrazione della sola routine in L.M. per consentire un suo impiego in programmi più complessi.

UN'IDEA PER CONCLUDERE

Questo "Speciale" ha avuto inizio da un'idea del mirabolante Ing. de Simone il quale aveva sempre cercato una routine che eliminasse gli errori dai dischetti in modo da adoperare, anche nell'ambito "otto bit", la procedura di protezione detta del "Laser Hole" (o foro laser), già in vigore per i sistemi MS-DOS.

Tecnicamente un foro su una superficie magnetica è visto dal sistema Drive come un errore su disco il quale, ovviamente, non è più correggibile dal momento che è di tipo hardware, cioè causato da un'alterazione fisica della superficie magnetica del dischetto.

La protezione suggerita fa semplicemente uso di questo brutale sistema per creare un errore che un copiatore, per quanto "magico" sia, può solo copiare come errore software e non più hardware.

Una routine elimina errori può svolgere il suo lavoro solo ed esclusivamente sugli errori software. Se, prima di andare a testare la presenza dell'errore in traccia "T", si fa operare sulla stessa traccia la routine "Elimina Errori", sul disco originale l'errore verrà trovato comunque, mentre, su quello copia, verrà eliminato dalla preventiva passata di cancellazione errori e pertanto non verrà trovato: semplice ma terribilmente efficace.

Se tale sistema di protezione viene combinato con un'efficace prevenzione anticrack per impedire di accedere alle routine di eliminazione e controllo errori, rappresenta sicuramente la vera ed unica protezione anticopia efficace al 99% con la quale non vi è copiatore presente nè futuro in

grado di creare copie funzionanti del software.

Chiaramente, per come è pensata, la protezione non è applicabile industrialmente (rigando un dischetto con uno spillo come si fa a sapere, dall'esterno, quali settori si stanno rovinando?) ma rappresenta sicuramente una valida proposta per la protezione di demo personali da inviare a Software House o da dare in pasto a personaggi potenzialmente pericolosi.

A titolo riassuntivo, ecco comunque la protezione così come è stata pensata da de Simone:

1. Formattare ex novo un dischetto
2. Con un laser, spillo, punteruolo etc. creare una micro scalfitura sul dischetto, in prossimità delle tracce più esterne (1, 2, etc.) oppure più interne (34, 35, etc.), cioè NON in mezzo al dischetto (traccia 18 e simili).
3. Con un programma del tipo "Disk Scanner" verificare l'intero disco per scoprire i settori distrutti con l'errore hardware ed il loro tipo.
4. Aggiungere, al programma da proteggere, due rotine da eseguirsi l'una dietro l'altra, PRIMA della partenza del programma vero e proprio: la prima deve tentare di eliminare gli errori presenti nei settori incriminati (dovrebbe, ovviamente, fallire nel tentativo); la seconda ha il compito di leggere gli stessi settori nel tentativo di rintracciare gli stessi errori che non dovrebbero essere stati "cancellati".
5. Registrare, sul dischetto così trattato, il programma da proteggere con incorporate le due routine (nasconderle bene!), avendo cura che non vada ad occupare nessun settore delle tracce danneggiate.

Al momento del caricamento e dell'esecuzione del programma, i casi possono essere due:

• Gli errori non si possono cancellare: il disco è quello originale ed il programma può "partire".
• Gli errori risultano eliminati: è quindi certo che il programma risiede su un disco - copia illegale: il programma provvederà, allora, a formattare il disco, a disallineare le testine, ad autocancellarsi, etc.

GLI ALTRI PROGRAMMI

Insieme a quelli già descritti, sul dischetto allegato vi sono programmi la cui utilità è strettamente connessa all'utilizzo del disk drive 1541.

Il loro impiego è solamente una delle funzioni a cui sono preposti; l'ampliamento e/o il miglioramento in generale delle routine proposte sono altrettanto importanti per impadronirsi al meglio delle tecniche di programmazione con le quali sono possibili molte caratteristiche inspiegabili come la verifica della presenza del Write Protect senza neanche fare accendere la luce del drive oppure la creazione di &-routine eseguibili nel 1541 con un'apparentemente innocente OPEN etc.

Ecco una breve spiegazione dei programmi presentati:

• Speedisk:
Programma didattico di un turbodisk con annesso file sorgente ultracommentato, per il macroassemblatore Merlin.

• Utility maker:
Programma di creazione di &-file come spiegato nell'articolo relativo.

• Disk Scanner:
Potentissimo programma di scansione di tutto (o parte) del dischetto per la visualizzazione dei parametri di Checksum & Id. per ciascun blocco, con l'analisi dell'eventuale presenza di errori.

• Drive monitor:
Se si sa già programmare in assembly ora lo si potrà fare direttamente all'interno della R.A.M. del disk drive 1541.

• Disk status:
Programma per la rilevazione dello stato del drive (canale di errore) in maniera del tutto automatica.

• Read block:
Programma di lettura di un settore dal disco.

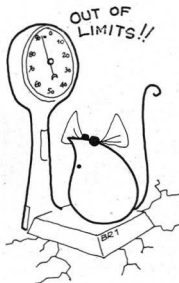
• Getspeed:
Lettore ultraveloce in L.M. di file sequenziali: con lo shift è possibile l'interruzione della lettura, che può terminare con il Run/Stop.

• Legge directory:
Un classico dal quale si può imparare molto sulla gestione (anche in assembly) delle comunicazioni con il disk drive.

• Blocchi liberi:
Un'altra della serie "piccole ma preziose"; le routine più semplici che quando servono in grossi programmi non si sa mai dove andare a cercare.

• Check W/P:
Utilissima routine che, senza far accendere il Led del drive, è in grado di stabilire se la protezione in scrittura è attivata con la presenza della linguetta di protezione sul dischetto o meno. Nella versione in L.M. viene usato il flag di Carry come risposta Si oppure No.

• Check Bad Sector:
Questa routine in linguaggio macchina richiede come parametri di ingresso la traccia ed il settore (registri X ed Y) in cui verificare la presenza di un errore il cui codice numerico viene consegnato nell'accumulatore; anche qui il flag di carry indicherà se l'errore da trovare è stato effettivamente riscontrato oppure no.

• Zoom disk:
Utility per caricare, visualizzare e salvare un qualsiasi blocco su dischetto; i comandi sono "R [T] [S]" per la lettura della traccia [T], settore [S], "W [T] [S]" per la scrittura degli stessi, "M [Inizio] [Fine]" per la visualizzazione di un blocco da [Inizio] a [Fine], "Chiocciolina" per la visualizzazione del canale di errore del drive, "Chiocciolina [Comando]" per l'invio del [Comando] al D.O.S. ed "X" per il ritorno al Basic: tutti i parametri devono essere specificati in esadecimale e la SYS di attivazione è la solita 49152.

```
DISASSEMBLATO COMMENTATO DI UNA ROUTINE DI CREAZIONE DEGLI ERRORI 20, 21, 22,
23, 27 e 29

"SYS" di attivazione: "M-E $0500" oppure "U3" oppure "UC" etc.

INIZIO ROUTINE:

.> 0400 20 A3 04  JSR $04A3    ; Setta la traccia ed il settore dell'intestazione
.> 0403 AD 15 04  LDA $0415    ; Numero dell'errore da 0 a 5
.> 0406 0A        ASL          ; Moltiplica per due
.> 0407 AA        TAX          ; Pone nel registro X
.> 0408 BD 16 04  LDA $0416,X  ; Valore basso del salto
.> 040B 85 BD     STA $BD      ; Lo pone in un puntatore di comodo
.> 040D BD 17 04  LDA $0417,X  ; Valore alto del Jump
.> 0410 85 BE     STA $BE      ; Lo pone in un puntatore di comodo
.> 0412 6C BD 00  JMP ($00BD)  ; Esegue la routine dell'errore richiesto

.> 0415 00        BRK          ; Locazione di flag contenente un numero da 0 a 5
indicante l'errore desiderato

.> 0416 30 04 ; $0430: Errore N.ro 20
.> 0418 00 07 ; $0700: Errore N.ro 21
.> 041A B7 04 ; $04B7: Errore N.ro 22
.> 041C C8 04 ; $04C8: Errore N.ro 23
.> 041E 22 04 ; $0422: Errore N.ro 27
.> 0420 2A 04 ; $042A: Errore N.ro 29


27, READ ERROR,TT,SS:
(Errore di chechsum nell'intestazione)

.> 0422 20 AC 04  JSR $04AC    ; Calcola la checksum dell'intestazione
.> 0425 38        SEC
.> 0426 E5 1A     SBC $1A      ; La sottrae dal valore corretto
.> 0428 85 1A     STA $1A      ; E la impone come checksum corrente


29, DISK ID MISMATCH,TT,SS:
(Errore di identificazione del disco)

.> 042A 20 00 06  JSR $0600    ; Modifica le intestazioni dei settori della
traccia da alterare
.> 042D 4C D7 04  JMP $04D7    ; Routine di chiusura della procedura


20, READ ERROR,TT,SS:
(Intestazione del blocco non trovata)

.> 0430 A9 15     LDA #$15
.> 0432 85 19     STA $19      ; Blocco dell'intestazione
.> 0434 20 34 F9  JSR $F934    ; Calcolo valori dell'header
.> 0437 A2 00     LDX #$00     ; X = 0
.> 0439 B5 24     LDA $24,X    ; Preleva gli otto valori d'intestazione
.> 043B 95 BD     STA $BD,X    ; E li "parcheggia" da $BD
.> 043D E8        INX
.> 043E E0 08     CPX #$08     ; Ripete per tutto l'header
.> 0440 D0 F7     BNE $0439
.> 0442 A9 55     LDA #$55     ; Valore per la modifica dell'intestazione
.> 0444 95 BD     STA $BD,X    ; Cambia l'header
.> 0446 95 BE     STA $BE,X    ; Cambia l'header
.> 0448 20 7A 04  JSR $047A    ; Lettura del blocco da modificare
.> 044B A9 FF     LDA #$FF     ; 0=IN, 1=OUT: $FF=%11111111=OUT
.> 044D 8D 03 1C  STA $1C03    ; Setta il registro direzione dati per la porta A
in uscita (OUT=WRITE)
.> 0450 AD 0C 1C  LDA $1C0C    ; Valore del PCR
.> 0453 29 1F     AND #$1F     ; Setta il Port Controller in uscita
.> 0455 09 C0     ORA #$C0     ; Abilita la scrittura
.> 0457 8D 0C 1C  STA $1C0C    ; PCR OUT
.> 045A B9 BD 00  LDA $00BD,Y  ; Carica il valore della "falsa" intestazione
.> 045D 8D 01 1C  STA $1C01    ; Lo scrive sul dischetto
.> 0460 B8        CLV          ; Dato scritto?
.> 0461 50 FE     BVC $0461    ; Attende il flag di scrittura
.> 0463 C8        INY          ; Prossimo dato
.> 0464 C0 0A     CPY #$0A     ; Fine dei valori dell'header?
.> 0466 D0 F2     BNE $045A    ; No, prosegui
.> 0468 50 FE     BVC $0468    ; Flag reset
.> 046A AD 0C 1C  LDA $1C0C    ; Valore del PCR
.> 046D 09 E0     ORA #$E0     ; Valore per il READ
.> 046F 8D 0C 1C  STA $1C0C    ; Port Controller IN
.> 0472 A9 00     LDA #$00     ; 0=IN, 1=OUT: $00=%00000000=IN
.> 0474 8D 03 1C  STA $1C03    ; Setta il registro direzione dati per la porta A
in lettura (IN=READ)
.> 0477 4C D7 04  JMP $04D7    ; Fine della procedura


ROUTINE DI LETTURA DEL BLOCCO DA MODIFICARE:

.> 047A A5 08     LDA $08      ; Traccia per il buffer 1 da $0400
.> 047C 85 18     STA $18      ; Come traccia dell'intestazione
.> 047E A6 09     LDX $09      ; Settore dell'intestazione
.> 0480 D0 04     BNE $0486    ; Salta se differente da zero
.> 0482 20 4B F2  JSR $F24B    ; Altrimenti stabilisce quanti settori ha la
traccia contenuta nell'accumulatore
.> 0485 AA        TAX          ; Pone in X il n. di settori massimo ( + 1 )
.> 0486 CA        DEX          ; Stabilisce l'ultimo settore valido nella
```

```
traccia: da zero ad "X"
.> 0487 86 19     STX $19      ; E lo conserva
.> 0489 20 12 05  JSR $0512    ; Legge e verifica l'intestazione
.> 048C 50 FE     BVC $048C    ; Attesa byte
.> 048E B8        CLV          ; Flag reset
.> 048F AD 01 1C  LDA $1C01    ; Legge il byte
.> 0492 C8        INY          ; Non memorizza il dato da nessuna parte ed
incrementa l'indice "Y"
.> 0493 D0 F7     BNE $048C    ; Ripete per la lettura dell'intero blocco
.> 0495 A0 BA     LDY #$BA     ; Puntatore per la lettura del gap
.> 0497 50 FE     BVC $0497    ; Byte pronto?
.> 0499 B8        CLV          ; Read flag reset
.> 049A AD 01 1C  LDA $1C01    ; Lettura del byte
.> 049D C8        INY          ; Nessuna memorizzazione: Y = Y + 1
.> 049E D0 F7     BNE $0497    ; Ripete per la lettura di tutti i valori di
"coda"
.> 04A0 4C 6B 05  JMP $056B    ; Riposizionamento al carattere "SYNC" di
sincronismo
```

ROUTINE DI MEMORIZZAZIONE DELLA TRACCIA & SETTORE ATTUALI COME TRACCIA & SETTORE DELL'INTESTAZIONE:

```
.> 04A3 A5 08     LDA $08      ; Traccia attuale
.> 04A5 85 18     STA $18      ; Lo pone come traccia d'intestazione
.> 04A7 A5 09     LDA $09      ; Settore attuale
.> 04A9 85 19     STA $19      ; Lo pone come settore d'intestazione
.> 04AB 60        RTS          ; Fine della routine
```

ROUTINE DI CALCOLO DELLA CHECKSUM D'INTESTAZIONE:

```
.> 04AC A9 00     LDA #$00     ; Valore iniziale della somma di controllo
.> 04AE 45 16     EOR $16      ; EOR Id.1
.> 04B0 45 17     EOR $17      ; EOR Id.2
.> 04B2 45 18     EOR $18      ; EOR traccia d'intestazione
.> 04B4 45 19     EOR $19      ; EOR settore d'intestazione
.> 04B6 60        RTS          ; Fine della routine
```

22, READ ERROR,TT,SS:
(Blocco dati non presente)

```
.> 04B7 20 83 05  JSR $0583    ; Lettura del settore da modificare
.> 04BA A9 08     LDA #$08     ; Valore di alterazione della costante per
l'inizio dei dati del blocco di testa, normalmente uguale ad otto
.> 04BC 85 47     STA $47      ; Impone il valore "illegale" 8 come costante
.> 04BE 20 A6 05  JSR $05A6    ; Riscrive il settore cosi' modificato
.> 04C1 A9 07     LDA #$07     ; Valore ooriginale della costante di inizio
.> 04C3 85 47     STA $47      ; Lo ripristina
.> 04C5 4C D7 04  JMP $04D7    ; Fine della procedura
```

23, READ ERROR,TT,SS:
(Errore nella checksum del blocco di dati)

```
.> 04C8 A5 3A     LDA $3A      ; Checksum attuale
.> 04CA 48        PHA          ; La conserva nello stack
.> 04CB 20 83 05  JSR $0583    ; Lettura del settore da alterare
.> 04CE 68        PLA          ; Recupera la vecchia somma di controllo
.> 04CF 85 3A     STA $3A      ; E la impone come quella attuale
.> 04D1 20 A6 05  JSR $05A6    ; Riscrive il settore cosi' alterato
.> 04D4 4C D7 04  JMP $04D7    ; Fine della procedura
```

ROUTINE DI TERMINE DELLA PROCEDURA DI CREAZIONE DEGLI ERRORI:

```
.> 04D7 A9 01     LDA #$01     ; Valore di flag per "tutto ok"
.> 04D9 A2 FF     LDX #$FF     ; Flag per fine procedura E.O.J. (End Of Job)
.> 04DB 86 51     STX $51      ; Settaggio del byte di flag
.> 04DD 4C 69 F9  JMP $F969    ; Routine di gestione degli errori Job Que
```

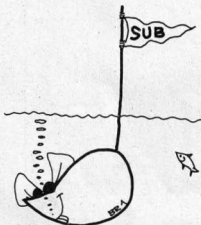
ROUTINE DI ESECUZIONE DELLA PROCEDURA DI CREAZIONE DEGLI ERRORI:

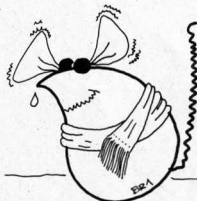
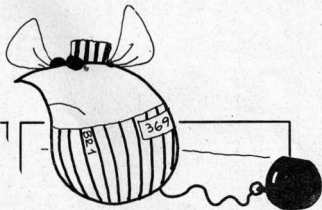
```
.> 0500 A2 03     LDX #$03     ; Numero dei tentativi in caso di errore
.> 0502 A9 E0     LDA #$E0     ; Comando Execute
.> 0504 85 01     STA $01      ; Esegue la routine nel buffer 1 da $0400
.> 0506 A5 01     LDA $01      ; Attende dalla routine d'interrupt il codice
d'errore
.> 0508 30 FC     BMI $0506    ; Se non e' pronto ( codice > 127 ) allora ripete
il controllo
.> 050A C9 01     CMP #$01     ; Errore uguale ad uno (tutto ok)?
.> 050C F0 03     BEQ $0511    ; Esce se affermativo
.> 050E CA        DEX          ; Prossimo tentativo
.> 050F D0 F1     BNE $0502    ; Riprova se i tentativi non sono esauriti
.> 0511 60        RTS          ; Fine dell'intera procedura
```

ROUTINE DI LETTURA DELL'INTESTAZIONE E DI RIPOSIZIONAMENTO ALL'INIZIO DELLA STESSA:

```
.> 0512 20 18 05  JSR $0518    ; Lettura dell'intestazione
.> 0515 4C 6B 05  JMP $056B    ; Rilegge il carattere di sincronismo
```

```
ROUTINE DI LETTURA HEADER:

.> 0518 20 34 F9  JSR $F934   ; Calcolo dei valori D'intestazione
.> 051B A5 25     LDA $25     ; Modifica i dati dell'header
.> 051D 29 C0     AND #$C0    ; Scrivendo valori scorretti
.> 051F 85 25     STA $25     ; E risistemandoli al posto dei dati originali
.> 0521 A5 26     LDA $26     ; Idem per un secondo dato dell'intestazione
.> 0523 29 0F     AND #$0F
.> 0525 85 26     STA $26
.> 0527 A2 5A     LDX #$5A    ; 90 tentativi
.> 0529 20 6B 05  JSR $056B   ; Attende il carattere di sincronismo dalla
testina
.> 052C A0 00     LDY #$00
.> 052E 50 FE     BVC $052E   ; Attende quindi la lettura del primo carattere
.> 0530 B8        CLV         ; resetta il flag di lettura
.> 0531 AD 01 1C  LDA $1C01   ; Legge il dato dalla testina
.> 0534 C5 24     CMP $24     ; E' uguale al dato in memoria?
.> 0536 D0 2B     BNE $0563   ; Salta se e' differente
.> 0538 50 FE     BVC $0538   ; Attende il byte dalla testina
.> 053A B8        CLV         ; Reset del flag di lettura
.> 053B AD 01 1C  LDA $1C01   ; Lettura del dato
.> 053E 29 C0     AND #$C0    ; Modifica secondo la maschera precedentemente
usata (Cfr. locazione $051D)
.> 0540 C5 25     CMP $25     ; Byte uguali?
.> 0542 D0 1F     BNE $0563   ; No, esci
.> 0544 50 FE     BVC $0544   ; Attesa byte
.> 0546 B8        CLV         ; Reset del flag di lettura
.> 0547 AD 01 1C  LDA $1C01   ; acquisizione dato dalla testina R/W
.> 054A 29 0F     AND #$0F    ; Modifica secondo la maschera precedentemente
usata (Cfr. locazione $0523)
.> 054C C5 26     CMP $26     ; Byte corretto?
.> 054E D0 13     BNE $0563   ; Esce se diverso
.> 0550 A0 00     LDY #$00    ; Indice Y a zero
.> 0552 50 FE     BVC $0552   ; Attesa byte
.> 0554 B8        CLV         ; Reset del read flag
.> 0555 AD 01 1C  LDA $1C01   ; byte letto
.> 0558 D9 27 00  CMP $0027,Y ; Confronto con i valori in memoria
.> 055B D0 06     BNE $0563   ; Esce se non sono uguali
.> 055D C8        INY         ; Prossimo valore da confrontare
.> 055E C0 02     CPY #$02    ; Ci sono altri byte?
.> 0560 D0 F0     BNE $0552   ; Si, prosegue il confronto
.> 0562 60        RTS         ; Fine della routine


ROUTINE DI CONTEGGIO DEL NUMERO DEI TENTATIVI DI LETTURA:

.> 0563 CA        DEX         ; Decrementa il numero dei tentativi
.> 0564 D0 C3     BNE $0529   ; Ripete se i tentativi non sono esauriti
.> 0566 A9 02     LDA #$02    ; Altrimenti il settore contiene gia' l'errore 20,
READ ERROR,tt,ss
.> 0568 4C D9 04  JMP $04D9   ; Fine della procedura


ROUTINE, ANALOGA ALL'EQUIVALENTE R.O.M. PRESENTE A PARTIRE DA $F556, DI ATTESA
PER IL CARATTERE DI SINCRONISMO:

.> 056B A9 FF     LDA #$FF    ; Valore di 255 per il timer
.> 056D 8D 05 18  STA $1805   ; Avvia il timer uno
.> 0570 A9 03     LDA #$03    ; Codice di errore "21, READ ERROR,tt,ss"
.> 0572 2C 05 18  BIT $1805   ; Il timer ha terminato la corsa?
.> 0575 10 F1     BPL $0568   ; Si, allora genera l'errore numero 21
.> 0577 2C 00 1C  BIT $1C00   ; Timer ancora valido: lettura del segnale di SYNC
.> 057A 30 F6     BMI $0572   ; Ripete se non ancora trovato
.> 057C AD 01 1C  LDA $1C01   ; SYNC valido, legge il byte
.> 057F B8        CLV         ; Ed azzera il flag di lettura ( e scrittura )
.> 0580 A0 00     LDY #$00    ; Azzeramento dell'indice "Y" per le operazioni da
e per il buffer
.> 0582 60        RTS         ; Termine della routine


ROUTINE SOSTITUTIVA DELLA R.O.M. PER LA LETTURA DI UN SETTORE:

.> 0583 A9 03     LDA #$03    ; Buffer da $0300 come deposito per la lettura
.> 0585 85 31     STA $31     ; Lo pone come buffer corrente per il disk
controller
.> 0587 20 12 05  JSR $0512   ; Lettura dell'intestazione
.> 058A 50 FE     BVC $058A   ; Attesa byte
.> 058C B8        CLV         ; Reset del flag R/W
.> 058D AD 01 1C  LDA $1C01   ; Legge il dato
.> 0590 91 30     STA ($30),Y ; Lo pone nel buffer corrente
.> 0592 C8        INY         ; Prossimo byte
.> 0593 D0 F5     BNE $058A   ; Prosegue a leggere 256 valori
.> 0595 A0 BA     LDY #$BA    ; Puntatore per i dati del gap di coda
.> 0597 50 FE     BVC $0597   ; Attesa dato
.> 0599 B8        CLV         ; Flag reset
.> 059A AD 01 1C  LDA $1C01   ; Lettura del byte
.> 059D 99 00 01  STA $0100,Y ; Lo pone nell'apposita area dello stack
.> 05A0 C8        INY         ; Prossimo valore
.> 05A1 D0 F4     BNE $0597   ; Legge tutti i valori del gap ponendoli da $01BA
a $01FF
.> 05A3 4C E0 F8  JMP $F8E0   ; Settaggio in memoria dei valori di chechsum,
Id., traccia, settore ed intestazioni lette
```

```
ROUTINE SOSTITUTIVA DELLA R.O.M. PER LA SCRITTURA DI UN SETTORE:

.> 05A6 A9 03     LDA #$03    ; Buffer da $0300 per il prelievo dei dati da
scrivere
.> 05A8 85 31     STA $31     ; Come buffer attuale del disk controller
.> 05AA 20 8F F7  JSR $F78F   ; Settaggio dei valori di checksum, Id., traccia e
settore
.> 05AD 20 18 05  JSR $0518   ; Lettura dell'intestazione per posizionare la
testina sul primo dato da sovrascrivere con il settore alterato
.> 05B0 A2 08     LDX #$08    ; Numero dei valori dell'intestazione
.> 05B2 50 FE     BVC $05B2   ; Attesa per il byte
.> 05B4 B8        CLV         ; Reset del flag e nessuna lettura del dato
.> 05B5 CA        DEX         ; Prossimo byte dell'intestazione
.> 05B6 D0 FA     BNE $05B2   ; Fine dei valori da "leggere"
.> 05B8 A9 FF     LDA #$FF    ; 0-IN, 1-OUT: $FF-%11111111-OUT
.> 05BA 8D 03 1C  STA $1C03   ; Setta il registro direzione dati per la porta A
in scrittura (OUT-WRITE)
.> 05BD AD 0C 1C  LDA $1C0C   ; Valore del PCR
.> 05C0 29 1F     AND #$1F    ; Setta il Port Controller in uscita
.> 05C2 09 C0     ORA #$C0    ; Abilita la scrittura
.> 05C4 8D 0C 1C  STA $1C0C   ; PCR OUT
.> 05C7 A9 FF     LDA #$FF    ; Falso valore di SYNC
.> 05C9 A2 05     LDX #$05    ; Per cinque volte
.> 05CB 8D 01 1C  STA $1C01   ; Lo scrive sul dischetto
.> 05CE B8        CLV         ; Reset del flag di scrittura
.> 05CF 50 FE     BVC $05CF   ; Attende il write flag
.> 05D1 B8        CLV         ; Lo resetta
.> 05D2 CA        DEX         ; Prossimo SYNC
.> 05D3 D0 FA     BNE $05CF   ; Prosegue se non ha ancora finito
.> 05D5 A0 BB     LDY #$BB    ; Puntatore per il gap
.> 05D7 B9 00 01  LDA $0100,Y ; Preleva dallo stack un valore del gap
.> 05DA 50 FE     BVC $05DA   ; Attende di poterlo scrivere
.> 05DC B8        CLV         ; Flag reset
.> 05DD 8D 01 1C  STA $1C01   ; Lo scrive
.> 05E0 C8        INY         ; Prossimo dato
.> 05E1 D0 F4     BNE $05D7   ; Prosegue sino al termine dei valori
.> 05E3 B1 30     LDA ($30),Y ; Prende un dato del blocco da scrivere
.> 05E5 50 FE     BVC $05E5   ; Attende per il flag di Ok Write
.> 05E7 B8        CLV         ; Reset del flag
.> 05E8 8D 01 1C  STA $1C01   ; Scrittura del byte
.> 05EB C8        INY         ; Prossimo dato
.> 05EC D0 F5     BNE $05E3   ; Salta se non e' stato scritto l'intero settore
.> 05EE 50 FE     BVC $05EE   ; Reset del flag R/W
.> 05F0 AD 0C 1C  LDA $1C0C   ; Valore del PCR
.> 05F3 09 E0     ORA #$E0    ; Valore per il READ
.> 05F5 8D 0C 1C  STA $1C0C   ; Port Controller IN
.> 05F8 A9 00     LDA #$00    ; 0-IN, 1- OUT: $00-%00000000-IN
.> 05FA 8D 03 1C  STA $1C03   ; Setta il registro direzione dati per la porta A
in ingresso (IN-READ)
.> 05FD 60        RTS         ; Fine della routine

ROUTINE DI MODIFICA DEI VALORI DELL'INTESTAZIONE:

.> 0600 20 A3 FD  JSR $FDA3   ; Scrive #$FF per 10240 volte
.> 0603 20 C3 FD  JSR $FDC3   ; Lettura dei SYNC scritti
.> 0606 A5 08     LDA $08     ; Traccia per il buffer 1 da $0400
.> 0608 20 4B F2  JSR $F24B   ; Stabilisce il numero massimo dei settori della
presente traccia
.> 060B 85 43     STA $43     ; Lo conserva nell'apposito byte in pagina zero:
numero massimo di settori della traccia attuale
.> 060D A0 00     LDY #$00    ; Y - 0
.> 060F 84 BD     STY $BD     ; Azzera il contatore dei settori gia' alterati
(byte di comodo $BD)
.> 0611 A5 39     LDA $39     ; Costante 8 per l'inizio dei dati del blocco di
testa
.> 0613 99 00 03  STA $0300,Y ; La deposita nel buffer
.> 0616 A5 BD     LDA $BD     ; Settore attuale
.> 0618 99 02 03  STA $0302,Y ; Lo deposita nel settore
.> 061B A5 08     LDA $08     ; Traccia
.> 061D 99 03 03  STA $0303,Y ; La deposita nel buffer
.> 0620 A5 13     LDA $13     ; Id.2 per il drive 0
.> 0622 99 04 03  STA $0304,Y ; Lo deposita nel buffer
.> 0625 A5 12     LDA $12     ; Id.1 per il drive 0
.> 0627 99 05 03  STA $0305,Y ; Lo deposita nel buffer
.> 062A A9 0F     LDA #$0F    ; Valore fittizio per la checksum
.> 062C 99 06 03  STA $0306,Y ; Lo pone nel buffer
.> 062F 99 07 03  STA $0307,Y ; Idem come sopra
.> 0632 A9 00     LDA #$00    ; Valore iniziale per la somma di controllo
.> 0634 59 02 03  EOR $0302,Y ; EOR settore
.> 0637 59 03 03  EOR $0303,Y ; EOR traccia
.> 063A 59 04 03  EOR $0304,Y ; EOR Id.2
.> 063D 59 05 03  EOR $0305,Y ; EOR Id.1
.> 0640 AE 15 04  LDX $0415   ; Numero dell'errore da creare ( da 0 a 5 )
.> 0643 E0 01     CPX #$04    ; E' l'errore numero 27?
.> 0645 D0 07     BNE $064E   ; Salta se non e' il 27 ( cioe' se e' il 29 )

SOLO PER L'ERRORE NUMERO 27:

.> 0647 18        CLC         ; Carry a zero
.> 0648 85 BE     STA $BE     ; Salva la checksum corretta in un byte di comodo
($BE)
.> 064A A5 1A     LDA $1A     ; Checksum precedente
.> 064C 65 BE     ADC $BE     ; La somma a quella corretta, generando cosi' una
somma di controllo errata
```

PROSEGUIMENTO DELLA ROUTINE COMUNE PER GLI ERRORI NUMERO 27 & 29:

```
.> 064E 99 01 03  STA $0301,Y ; Conserva la checksum cosi' ottenuta
.> 0651 18        CLC         ; Azzera il riporto
.> 0652 98        TYA         ; Indice del buffer da $0300 in accumulatore
.> 0653 69 08     ADC #$08    ; numero di bytes occupati nel buffer da $0300 per
ciascuna intestazione della traccia da alterare
.> 0655 A8        TAY         ; Setta l'indice "Y" al nuovo valore
.> 0656 E6 BD     INC $BD     ; Incrementa il contatore dei settori gia'
"trattati"
.> 0658 A5 BD     LDA $BD
.> 065A C5 43     CMP $43     ; Si tratta dell'ultimo settore della traccia?
.> 065C 90 B3     BCC $0611   ; Non ancora, prosegue nella modifica dei
rimanenti blocchi
.> 065E A9 03     LDA #$03    ; Buffer da $0300
.> 0660 85 31     STA $31     ; Lo pone come buffer dei dati per il disk
controller
.> 0662 98        TYA .       ; Trasferisce nell'accumulatore il puntatore nel
buffer da $0300
.> 0663 48        PHA         ; E lo conserva nello stack
.> 0664 20 30 FE  JSR $FE30   ; Trasferisce nelle relative locazioni gli Id. e
le somme di controllo calcolate
.> 0667 68        PLA         ; Recupera il puntatore nel buffer
.> 0668 A8        TAY         ; Lo ripristina nel registro "Y"
.> 0669 88        DEY         ; Decrementa l'indice
.> 066A 20 E5 FD  JSR $FDE5   ; E trasla di $45 bytes in avanti il contenuto del
buffer 0 da $0300
.> 066D 20 F5 FD  JSR $FDF5   ; Copia i valori da $01BB a $01FF ponendoli da
$0300 a $0344
.> 0670 20 0E FE  JSR $FE0E   ; Scrive #$55 ( = %01010101 ) per 10240 volte
.> 0673 A9 00     LDA #$00    ; Accumulatore = 0
.> 0675 85 32     STA $32     ; Reset del puntatore basso del buffer attuale
.> 0677 A9 FF     LDA #$FF    ; Valore %11111111
.> 0679 8D 01 1C  STA $1C01   ; Scrittura sul dischetto
.> 067C A2 05     LDX #$05    ; Numero delle scritture di #$FF
.> 067E 50 FE     BVC $067E   ; Write Ok?
.> 0680 B8        CLV         ; Si, reset del flag relativo
.> 0681 CA        DEX         ; #$FF scritto per cinque volte?
.> 0682 D0 FA     BNE $067E   ; Ripete se negativo
.> 0684 A2 0A     LDX #$0A    ; Dieci volte
.> 0686 A4 32     LDY $32     ; Puntatore nel buffer
.> 0688 50 FE     BVC $0688   ; Attende la scrittura
.> 068A B8        CLV         ; Azzera il flag di overflow
.> 068B B9 00 03  LDA $0300,Y ; Lettura di un byte dal buffer
.> 068E 8D 01 1C  STA $1C01   ; E scrittura sul dischetto
.> 0691 C8        INY         ; Y = Y + 1
.> 0692 E6 32     INC $32     ; Incremento del puntatore nel buffer
.> 0694 CA        DEX         ; Gia' scritti dieci valori?
.> 0695 D0 F1     BNE $0688   ; No, ripeti
.> 0697 A2 08     LDX #$08    ; Otto volte
.> 0699 50 FE     BVC $0699   ; Byte pronto?
.> 069B B8        CLV         ; Ok, reset flag
.> 069C A9 55     LDA #$55    ; SYNC #$55=%01010101
.> 069E 8D 01 1C  STA $1C01   ; Scrittura del carattere di sincronismo
.> 06A1 CA        DEX         ; Gia' scritti otto valori?
.> 06A2 D0 F5     BNE $0699   ; Ripete se negativo
.> 06A4 A9 FF     LDA #$FF    ; Valore #$FF=%11111111'
.> 06A6 A2 05     LDX #$05    ; Cinque volte
.> 06A8 50 FE     BVC $06A8   ; Attesa per il byte pronto
.> 06AA B8        CLV         ; Azzera il flag di overflow
.> 06AB 8D 01 1C  STA $1C01   ; Alla testina di scrittura
.> 06AE CA        DEX         ; Gia' scritti cinque valori?
.> 06AF D0 F7     BNE $06A8   ; No, ripete
.> 06B1 A9 55     LDA #$55    ; Valore #$55=%01010101 in sostituzione dei valori
di gap da $01BB a $01FF
.> 06B3 A2 BB     LDX #$BB    ; Pseudo-puntatore nello stack per i valori di gap
in sostituzione dell'originale istruzione R.O.M. "LDA $0100,X"
.> 06B5 50 FE     BVC $06B5   ; Byte pronto?
.> 06B7 B8        CLV         ; Reset del flag di R/W
.> 06B8 8D 01 1C  STA $1C01   ; Scrittura di %01010101
.> 06BB E8        INX         ; Prossimo valore
.> 06BC D0 F7     BNE $06B5   ; Ripete se non ha terminato
.> 06BE A2 00     LDX #$00    ; X = 0
.> 06C0 50 FE     BVC $06C0   ; Byte pronto?
.> 06C2 B8        CLV         ; Ok, flag reset
.> 06C3 8D 01 1C  STA $1C01   ; Scrittura di ulteriori 256 valori
.> 06C6 E8        INX         ; X = X + 1
.> 06C7 D0 F7     BNE $06C0   ; Ripete se non ha terminato
.> 06C9 A2 08     LDX #$08    ; Altre otto volte
.> 06CB 50 FE     BVC $06CB   ; Byte pronto?
.> 06CD B8        CLV         ; Azzeramento del R/W flag
.> 06CE 8D 01 1C  STA $1C01   ; Scrittura del dato
.> 06D1 CA        DEX         ; Altri valori
.> 06D2 D0 F7     BNE $06CB   ; Se affermativo, ripete
.> 06D4 C6 BD     DEC $BD     ; Ripete per tutti i settori della traccia
.> 06D6 D0 9F     BNE $0677   ; Salta se si devono alterare altri settori
.> 06D8 50 FE     BVC $06D8   ; Attesa byte
.> 06DA B8        CLV         ; Reset flag
.> 06DB 20 00 FE  JSR $FE00   ; Setta il Port Controller in lettura (Input)
.> 06DE 4C D7 04  JMP $04D7   ; Fine della procedura
```

```
21, READ ERROR,TT,SS:
(Nessun carattere di sincronizzazione)

.> 0700 20 7A 04  JSR $047A   ; Lettura del blocco da modificare
.> 0703 A9 FF     LDA #$FF    ; 0-IN, 1-OUT: $FF-%11111111-OUT
.> 0705 8D 03 1C  STA $1C03   ; Setta il registro direzione dati per la porta A
in uscita (OUT-WRITE)
.> 0708 AD 0C 1C  LDA $1C0C   ; Valore del PCR
.> 070B 29 1F     AND #$1F    ; Setta il Port Controller in uscita
.> 070D 09 C0     ORA #$C0    ; Abilita la scrittura
.> 070F 8D 0C 1C  STA $1C0C   ; PCR OUT
.> 0712 A9 00     LDA #$00    ; Valore basso del contatore dei SYNC
.> 0714 85 F0     STA $F0     ; Lo pone in un byte di comodo
.> 0716 A9 00     LDA #$00    ; Valore alto del contatore dei SYNC
.> 0718 85 F1     STA $F1     ; Lo pone in un byte di comodo
.> 071A A9 55     LDA #$55    ; SYNC $55-%01010101
.> 071C 8D 01 1C  STA $1C01   ; Scrive il SYNC
.> 071F B8        CLV         ; Write flag reset
.> 0720 50 FE     BVC $0720   ; Dato scritto?
.> 0722 C6 F0     DEC $F0     ; Diminuisce il contatore low
.> 0724 D0 F4     BNE $071A   ; Ripete se non ha ancora finito
.> 0726 C6 F1     DEC $F1     ; Decrementa il contatore high
.> 0728 D0 F0     BNE $071A   ; Ripete se non ha ancora finito
.> 072A AD 0C 1C  LDA $1C0C   ; Valore del PCR
.> 072D 09 E0     ORA #$E0    ; Abilita la lettura
.> 072F 8D 0C 1C  STA $1C0C   ; Port Controller IN
.> 0732 A9 00     LDA #$00    ; 0-IN,1-OUT: $00-%00000000-IN
.> 0734 8D 03 1C  STA $1C03   ; Setta il registro direzione dati per la porta A
in input (IN-READ)
.> 0737 4C D7 04  JMP $04D7   ; Fine della procedura
```

TUTTI GLI ERRORI DEL 1541

Come interpretare i messaggi di errore emessi dal 1541 per scoprire che cosa si nasconde dietro il lampeggio del Led

Tra le caratteristiche intelligenti dell'unità a dischi Commodore, alla quale è dedicato, il presente fascicolo, particolarmente gradita risulta essere la capacità di emettere alcuni messaggi di errore i quali, oltre ad un codice numerico associato, sono anche costituiti da una breve descrizione che molto spesso è sufficiente ad identificare con precisione l'errore eventualmente verificatosi durante l'operazione di lettura o scrittura.

Dall'esterno del drive un errore è direttamente rilevabile dal continuo lampeggiare di un Led il quale, normalmente, si accende di luce continua durante le normali operazioni di Read & Write e rimane spento nei momenti di inattività della testina R/W.

Essendo il drive un dispositivo esterno, non può visualizzare nulla sul video dell'utente (sia esso monitor oppure televisore) in quanto la gestione della visualizzazione è interamente affidata al computer centrale, con particolare riferimento agli indirizzi di memoria da 1024 ($400) a 2023 ($7E7) del Commodore 64.

Il Led lampeggiante, pertanto, è tutto quanto il drive può fare in attesa che il messaggio di errore occorso (la cui descrizione è intanto stata preparata dal 1541) venga "prelevato" attraverso un canale che altro non poteva essere se non quello dei comandi con associato il solito indirizzo secondario 15.

Prima di "estrarre" dal drive la preziosa informazione sarà opportuno esplorare il formato del dischetto e la sintassi costituente il messaggio di errore (e relativa semantica dello stesso) per meglio comprendere come e perchè si può generare un errore nel 1541.

IL FORMATO DEL DISCHETTO PER IL 1541

I dischetti usati dal drive sono del tipo a singola faccia (nel quale, cioè, i dati vengono scritti e letti da un solo lato del dischetto, che è quello inferiore) ed hanno 35 tracce concentriche come.

La superficie magnetica delle tracce è a sua volta suddivisa in un numero variabile di settori, chiamati anche blocchi, ciascuno dei quali è composto da un numero variabile di byte, la cui parte più "succosa" ed utile è sempre lunga 256 byte, generalmente composti dai primi due di collegamento tra tutti i blocchi di un file (rispettivamente traccia e settore successivi) e da 254 byte di dati.

Per la loro scrittura sul dischetto viene impiegato un particolare formato, denominato G.C.R. (Group Code Recording = registrazione di codice di gruppo) a causa della modalità di rilevazione dei valori da parte della testina del drive. Questa può solo verificare la presenza di un bit ad 1 e, indirettamente, un bit a 0 attraverso l'assenza di un bit ad 1; tale formato è, infatti, l'unico impiegato anche in tutti gli altri D.O.S. delle unità a dischi C.B.M. cronologicamente anteriori al 1541.

Nella figura riportata in queste pagine viene ingrandito un settore qualsiasi per comprendere come il D.O.S. codifichi e trascriva i dati sul supporto magnetico nei vari blocchi, unità di misura ufficiale dello spazio occupato sui floppy disk del 1541.

La prima sigla riportata è detta "SYNC 1" ed indica una sequenza di 40 bit posti ad uno (cioè 5 byte a #$FF) come carattere di sincronismo per la frequenza di scrittura massima identificata da una successione di variazioni dello stato magnetico, che la testina R/W interpreta come dei bit ad 1; lo stato 0 del bit lo si ricava, invece, in conseguenza di una mancanza di variazione nell'orientamento delle particelle magnetiche dello stesso supporto.

Orbene, questo sincronismo viene normalmente sfruttato dal D.O.S. come conferma dell'esatto posizionamento della testina nel punto dal quale potranno essere lette, o scritte, le informazioni richieste. Queste, come si vedrà tra breve, non sono costituite solo ed esclusivamente dai 256 byte del settore in oggetto.

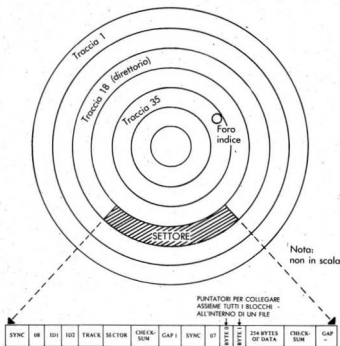
Di seguito ai 5 byte #$FF si trova una costante, solitamente contenuta nella locazione 57 ($39) del disk drive, che ha valore standard di 8: indica l'inizio dei dati del blocco di testa (attenzione: non dei dati!) nel quale sono contenute le indicazioni "geografiche" di dove ci si trova nell'ambito del floppy disk oltre ad un'indicazione di identificazione del dischetto.

Di seguito al valore #$08, infatti, si trovano 5 byte dei quali gli ultimi quattro rappresentano, rispettivamente, il numero di settore del blocco ove ci si trova ("SK"), il suo numero di traccia ("TK"), il secondo ("Id.2") ed il primo ("Id.1") dei caratteri di identificazione specificati all'atto della formattazione del disco.

Il primo dei 5 byte, siglato "CHK 1", rappresenta un valore di checksum (in italiano equivale a "somma di controllo") impiegato dall'onnipresente D.O.S. per stabilire se i quattro valori successivi ("SK", "TK", "Id.2" ed "Id.1") sono corretti oppure se qualche dato risulta alterato rendendo di fatto problematica l'identificazione univoca del settore richiesto.

All'atto della registrazione dell'intestazione (denominata anche "header", in inglese) viene infatti calcolato il valore di checksum "CHK 1" con una routine R.O.M. del 1541, presente a partire da $FC69 la quale opera una serie di EOR (istruzione assembly di OR esclusivo) tra i quattro valori citati precedentemente, a partire dall'accumulatore posto uguale a zero (valore iniziale della somma di controllo) ed il risultato finale della procedura viene conservato come "CHK 1".

Proseguendo tra le successive etichette si incontra la "GAP 1" dietro la quale si "nascondono" ben 72 bit nei quali si alternano gli stati magnetici 0 ed 1; il tutto si traduce in 9 byte contenenti il valore #$55 (binario %01010101) che fungono da separatori ("gap", infatti, sta per "spazio vuoto" in una zona del supporto magnetico) tra l'header del blocco di testa (il già visto "SK", "TK", etc.) e quello dei dati effetti-

APPENDICE D: FORMATI SUL FLOPPY DISK

Formato del 1540/1541: espansione di un settore

vi del blocco i quali si incontreranno più avanti.

Svolgendo le funzioni analoghe a quelle del "SYNC 1", si trova di seguito la sigla "SYNC 2" formata dai soliti 40 bit posti ad 1 (5 byte contenenti il valore #$FF) e quindi dal valore #$07, solitamente contenuto nella locazione 71 ($47) della pagina zero del 1541; le funzioni sono simili a quelle svolte in precedenza dalla costante 8: è un indicatore di identificazione del campo dei dati del blocco cercato.

"BYTE 0" e "BYTE 1" sono (finalmente!) i primi due byte di dati effettivi del settore (in pratica quelli leggibili dai comandi tipo "U1", "B-R", il LOAD stesso, etc.) che hanno una propria etichetta per la funzione svolta in caso di registrazione di un blocco di un file: contengono (come già dovrebbe essere noto) la traccia ("BYTE 0") ed il settore ("BYTE 1") dal quale il D.O.S. continuerà la lettura del file.

I 254 byte successivi sono il complemento a 256 che forma l'effettivo blocco dei dati memorizzati sul dischetto. In seguito si trova la "CHK 2" che viene generata con l'EOR tra i 256 byte di dati ("BYTE 0" + "BYTE 1" + "DATI") ad opera di una semplicissima routine (sempre R.O.M.) presente nel 1541 da $F5E9 che opera secondo le stesse modalità specificate in occasione della già citata routine di "CHK 1".

Conclude il blocco un "GAP 2" il quale risulta esere formato da ulteriori 72 bit 0101... rilevabili nella forma più comprensibile di 9 byte aventi sempre il valore #$55; tale indicazione è però valida solo per i settori intermedi, mentre tra l'ultimo blocco di una traccia ed il primo (dove "ricomincia" la traccia stessa), il numero dei byte di "GAP 2" è variabile in maniera inversamente proporzionale alla velocità di rotazione del floppy disk nel drive che dovrebbe (ma non lo è mai) essere uguale a 5 giri al secondo, 300 per minuto primo.

IL MESSAGGIO DI ERRORE

Si osserverà, a questo punto, il formato attraverso il quale viene consegnato il messaggio di errore sul canale 15.

Sostanzialmente è formato da 3 "campi" numerici e da un campo stringa, formato cioè da caratteri che possono essere lettere o numeri.

Le quattro indicazioni sono fornite di seguito, con il carattere di virgola CHR$(44) (esadecimale $2C) come separatore tra i campi.

Il primo di questi contiene un codice numerico compreso tra 20 e 74 in caso di segnalazione di errore, oppure contiene 0 od 1 in caso di messaggi non di errore.

Dopo il carattere di separazione si trova il campo stringa alfanumerico contenente una breve descrizione dell'errore numerico precedente la virgola e, quindi, gli ultimi due campi numerici con l'indicazione, rispettivamente, di traccia e di settore nel quale l'errore ha avuto luogo.

Un esempio di messaggio di errore emesso dal drive, sicuramente occorso a chiunque ne abbia usato uno, è l'errore numero 62 in occasione della mancata individuazione di un file del quale si è chiesta l'apertura in lettura (oppure in append). Da Basic basta chiedere il LOAD di un file che non esiste e prontamente il blink del Led non si farà attendere.

Leggendo il canale di errore (se non lo si sa già, tra breve si imparerà a farlo) si otterrà...

62, FILE NOT FOUND, 00, 00

...messaggio che sui Commodore 16, Plus/4 e C/128 è ottenibile con il comando...

PRINT DS$

...mentre su un C/64 dovrà essere letto tramite programma apposito.

Ricordandosi dell'uso della poco impiegata istruzione basic INPUT#, si può ad essa ricorrere per leggere i quattro campi, separati da virgole, del messaggio di errore:

```
90 REM ROUTINE DI LETTURA DEL CANALE DI ERRORE
100 OPEN 1, 8, 15
110 INPUT# 1, E, M$, T, S
120 CLOSE 1
130 PRINT E; M$; T; S
140 END: REM RETURN
```

La procedura, quindi, prevede una semplice lettura del canale di errore, che, in alternativa, può anche essere svolta da altra routine che ricorre, però, all'istruzione basic GET# piuttosto che ad INPUT#:

```
90 SECONDA ROUTINE DI LETTURA DEL
CANALE DI ERRORE
100 OPEN 1, 8, 15
110 GET# 1, A$
120 PRINT A$;
130 IF ST=64 THEN 140
140 GOTO 110
140 CLOSE 1
150 END: REM RETURN
```

Questa seconda routine svolge l'identica funzione della variabile riservata DS$ dei Commodore 16, Plus/4 e C/128, dal momento che stampa sul video tutti i caratteri provenienti dal 1541, virgole comprese.

Da notare un'applicazione pratica nell'uso del bit 6 (valore = 64) della locazione 144 ($90 = variabile STatus) che varrà 1 (ponendo quindi ST=64) se viene raggiunta la fine del file, in questo caso costituito dalla sequenza di caratteri formanti il messaggio di errore.

Nel caso di FILE NOT FOUND (da non confondersi, si badi bene, con il messaggio di errore del basic il quale non ha bisogno di essere richiamato dal drive poichè viene generato dal computer e non ha le indicazioni numeriche dei numeri di errore, traccia e settore), i numeri di traccia e settore sono entrambi posti a zero per indicare che l'errore non è occorso in un blocco specifico del disco, ma si tratta di un errore logistico di carattere generale: se un file non esiste sul floppy disk non è corretto accusare un settore particolare di non contenerlo!

Precedentemente si è parlato dei numeri di errore 0 ed 1 i quali non indicano un messaggio di errore propriamete detto. Il primo ha "OK" come descrizione e "00, 00" come numeri di traccia e settore: si intuisce piuttosto facilmente la sua funzione che è quella di comunicare che tutto va bene e non vi sono errori in corso.

L'errore numero 1 ha la descrizione "FILES SCRATCHED" ed ha luogo subito dopo aver impartito il comando "S" di Scratch con il quale è possibile cancellare i file presenti sul dischetto.

Il comando Scratch ha la possibilità della multiselezione con la quale è possibile cancellare più files con un comando solo di "S". In seguito alla generazione del messaggio numero 01 di FILES SCRATCHED, il campo che normalmente dovrebbe contenere la traccia dell'errore (subito dopo il campo stringa di descrizione dell'errore stesso) conterrà il numero dei file effettivamente cancellati dall'ultimo comando di Scratch impartito. Per esempio...

01, FILES SCRATCHED, 00, 00

...indica che il nome specificato nel comando "S" non esiste sul dischetto e, pertanto, il numero dei file cancellati risulta essere zero.

LA CODIFICA DEI MESSAGGI DI ERRORE

Il D.O.S. del disk drive 1541 è in grado di emettere numerosi messaggi di errore: eccone le cause ed i possibili rimedi

Molti tra i dischetti dei programmi normalmente presenti sul mercato vengono creati con l'intenzionale presenza di qualche errore, allo scopo di impedirne la duplicazione abusiva.

Tendenzialmente questa consuetudine è scaturita dallo sfruttamento delle caratteristiche del D.O.S. il quale, come si è già avuto modo di osservare in precedenza, effettua una lunga serie di verifiche (ricerca del carattere di sincronismo, verifica della somma di controllo,
etc.) in seguito alle quali si rifiuta di leggere il blocco se incontra qualche dato diverso da quello che avrebbe dovuto ottenere.

Questa, quindi, è senz'altro una delle cause più frequenti per le quali un drive inizia a lampeggiare durante il caricamento di programmi delle software-house.

A parte l'errata digitazione del nome, infatti, un drive ben difficilmente presenta problemi nell'uso normale, anche se può capitare una smagnetizzazione involontaria del disco che modifichi qualche carattere nell'intestazione (indicazioni di traccia, settore ed Id.), prontamente rilevata dalla chechsum 1 che conterrà un valore differente da quello ricalcolato ad ogni Read, oppure un'alterazione di qualche byte di dati, per il quale la "CHK 2" farà buona guardia.

Tutto è sotto controllo, insomma, ma nello sfortunato caso in cui si scopra un'involontaria modifica dello stato magnetico del floppy disk, qui di seguito sono riportati tutti i messaggi di errore generabili dal disk drive 1541.

00, OK, 00, 00
(Nessun errore)

Messaggio tra i più frequenti quando si controlla il canale degli errori. Indica che non sussiste alcun errore corrente nel drive.

01, FILES SCRATCHED, nn, 00
Informazioni sul numero di file cancellati. Messaggio generato in seguito ad un comando di scratch che indica l'avvenuta cancellazione di "nn" file.

02 / 19
Non esistono messaggi di errore individuati da questi valori

20, READ ERROR, tt, ss
Intestazione del blocco non trovata. Il D.O.S. non riesce a localizzare l'intestazione del blocco dati richiesto in traccia "tt" e nel settore "ss". Generalmente significa che l'intestazione è stata cancellata oppure che la testina R/W gestita dal controller del disco (locazioni particolari che controllano i motori per lo spostamento della testina presenti nel 1541 nelle locazioni \$1800 / \$1C00 e successive) è stata indirizzata ad un blocco illegale.

21, READ ERROR, tt, ss
Mancanza del carattere di sincronizzazione. Anche questo è un "Read Error" (errore di lettura) ed ha luogo quando sulla traccia richiesta il D.O.S. (nella fattispecie, il controller del disco) non riesce a trovare il carattere di sincronizzazione. Nei casi peggiori l'errore 21 può indicare problemi a livello hardware (se, ad esempio, persiste per tutti i floppy inseriti nel drive); di solito, invece, le cause sono le seguenti:
- Dischetto non inserito correttamente nel disk drive o assente
- Dischetto non formattato nella maniera corretta oppure non formattato del tutto.
- Mancanza di allineamento, in generale, tra la testina R/W e la traccia desiderata

22, READ ERROR, tt, ss
Blocco dei dati non presente. Al controller dell'unità a dischi è stata richieta la lettura, oppure la verifica, di un blocco dati in precedenza scritto in maniera non corretta. Molto spesso tale errore ha luogo dopo i comandi di Block ed indica una richiesta da parte degli stessi di un settore illegale.

23, READ ERROR, tt, ss
Errore di chechsum nel blocco dati. Tra gli errori è uno dei relativamente meno gravi in quanto fa semplicemente sapere che è stato letto il blocco richiesto ed il problema risiede solo nella somma di controllo "CHK 2" la quale non ha il valore che dovrebbe avere. Se l'errore non ha origini hardware (messa a terra e simili) ed i dati sono corretti, è sufficiente la riscrittura del blocco per eliminare il problema: la corretta checksum verrà ricalcolata e riscritta insieme ai dati del settore.

24, READ ERROR, tt, ss
Errore di decodifica dei byte. Questo messaggio di errore, sul quale dubito che occhio umano possa mai essersi posato, informa che i dati (intestazione compresa) sono stati letti nella memoria dell'unità a dischi, ma ha avuto luogo un errore hardware da una configurazione non valida nel byte di dati, generalmente provocato da problemi di messa a terra a causa di masse scollegate ed analoghe amenità.

25, WRITE ERROR, tt, ss
Errore di verifica in scrittura. Il primo dei messaggi di errore in scrittura viene generato se, durante la scrittura di un blocco, il confronto tra il dato effettivamente scritto, e quello presente in memoria, segnala qualche differenza durante la fase di verifica. La causa più comune è l'uso recidivo di quei dischetti che si trovano in regalo nelle patatine o che si ricevono in omaggio dalle bancarelle acquistando un chilogrammo di caldarroste: la soluzione è quella di ricorrere ad un altro dischetto, possibilmente di alta qualità.

26, WRITE PROTECT ON, tt, ss
Protezione in scrittura attivata. Inserendo un floppy disk nel drive, si sarà notato che sul lato sinistro è presente una tacca la quale dovrebbe rimanere coperta da un'apposita linguetta se non si ha intezione di scrivere o modificare qualcosa sul dischetto. Così come nelle compact cassette vi sono delle linguette che proteggono il nastro da eventuali (indesiderate) sovrascritture, allo stesso modo la finestrella vi-

sta sul dischetto permette di scrivere e cancellare qualcosa solamente se priva della linguetta di protezione. Nel caso in cui si tentino le stesse operazioni dopo aver coperto la finestrella in questione, si potrà ammirare in tutta la sua beltà il simpatico errore numero 26.

27, READ ERROR, tt, ss
Errore di checksum nell'intestazione del blocco. Dal momento che l'intestazione del blocco (contenente, lo si ricorda nuovamente, "SK", "TK", "Id.2" ed "Id.1") viene scritta prima del blocco dei dati vero e proprio, in seguito al verificarsi di tale errore (valore errato di "CHK 1") il blocco non verrà neanche letto nella memoria del 1541, a differenza di quanto avviene con l'errore numero 23 inerente alla "CHK 2". L'usuale causa è dovuta ad un problema di cattivo isolamento dell'unità, messa a terra difettosa, etc.

28, WRITE ERROR, tt, ss
Blocco di dati troppo lungo. Il controller dell'unità a dischi cerca di rilevare il carattere di sincronismo dell'intestazione successiva, dopo aver scritto un blocco di dati. Se la sincronizzazione non avviene entro il tempo previsto (scandito dai timer interni del 1541) avverrà la generazione del messaggio di errore. Scartando il (peraltro possibile) difetto di tipo hardware, l'errore può essere causato da un formato particolare (comunque non standard) del dischetto, dal momento che i dati si estendono sul blocco successivo.

29, DISK ID MISMATCH, tt, ss
Errore di identificazione del floppy disk. Ciascun blocco del supporto magnetico possiede, come già visto, una preziosissima intestazione nella quale, tra le altre, sono contenute le indicazioni dei due caratteri di identificazione del dischetto ("Id. 2" ed "Id. 1") specificati all'atto della formattazione dello stesso. Prima di una qualunque operazione il D.O.S. controlla se i valori interni di Id. (caricati con il comando di "Initialize") corrispondono a quelli del dischetto attualmente presente nel disk drive: in caso di discordanza si ha l'emissione dell'errore. Se si opera con comandi "tranquilli" tipo il LOAD etc, l'Initialize viene inviato automaticamente; se si opera con i comandi ad accesso diretto, e si vuole evitare l'errore, ci si deve ricordare di aprire sempre la comunicazione inviando un comando di "I" come prima operazione dopo la OPEN; esempio:

OPEN 1, 8, 15, "I"

Se non causato da una mancata inizializzazione del dischetto, tale errore può anche essere dovuto all'effettiva presenza di coppie di caratteri di Id. differenti sullo stesso dischetto provocati da una smagnetizzazione involontaria (o meno...).

30, SYNTAX ERROR, tt, ss
Errore generale di sintassi in un comando inviato all'unità a dischi. Il più conosciuto dei messaggi di errore basic è presente anche nel 1541 e, probabilmente, si avrà modo di sorbirselo parecchio durante i primi esperimenti di invio dei comandi al 1541: nessun timore: è tutto regolare e basterà un minimo d'esperienza per dimenticarlo del tutto. Infatti la motivazione dell'errore numero 30 è essenzialmente dovuta all'impossibilità, da parte del Disk Operating System, di interpretare il comando inviato sul canale. Le cause più frequenti vanno ricercate tra qualche configurazione errata del comando, oppure da un numero non corretto di nomi di file specificati come, per esempio, un comando "R" di Rename senza il carattere di uguale seguito dal nuovo nome che si desidera assegnare al file. L'unica soluzione a questo errore è quella di inviare nuovamente il comando, controllando attentamente la sintassi dello stesso.

31, SYNTAX ERROR, tt, ss
Comando non valido. Nell'invio dei comandi al disk drive è opportuno tenere presente che la lettera che lo identifica deve sempre occupare la prima posizione nell'ambito della stringa inviata all'unità a dischi; il rimedio è quello di controllare i caratteri da inviare prima di ripetere il comando.

32, SYNTAX ERROR, tt, ss
Stringa di comando troppo lunga. Durante l'esposizione dei vari comandi si è sempre riportata la forma abbreviata degli stessi, generalmente formata dalle sole iniziali del comando come, ad esempio, "I" per "Initialize". Tuttavia il D.O.S. consente anche la scrittura per esteso di questo (e di altri) comandi, per poter così far ammirare all'utente quanto simpatico sia l'errore numero 32. Tale limitazione è sostanzialmente dovuta alle non infinite dimensioni del buffer di input del 1541 il quale si estende (si fa per dire...) dalla locazione 512 ($200) alla 552 ($228).

Al di là di tale lunghezza il D.O.S. è costretto a sovrascrivere le tabelle di associazione dei canali presenti negli indirizzi successivi; esagerando con il numero di caratteri inviati in una volta sola, si obbligherà il 1541 a preparare il messaggio di SYNTAX ERROR numero 32 mentre il Led inizierà il suo sfogo personale a base di ON ed OFF ripetuti.

33, SYNTAX ERROR, tt, ss
Nome di file non valido. Se da un lato la comodità dei caratteri di configurazione (l'asterisco "*" ed il punto interrogativo "?") è di indubbia utilità nell'indicare un nome di file del quale non si rammentano uno o più caratteri (esempio: "C?A*" che può significare "CIAO", "CRANIO", etc.), dall'altro il D.O.S. "spera" che, almeno nella fase di SAVE o, più genericamente, dell'apertura di un file in scrittura, l'utente si ricordi il nome esatto da assegnare al file, dal momento che lo sta creando or ora. La soluzione per evitare il messaggio di errore numero 33, pertanto, è quella di riscrivere correttamente il nome del file, evitando di adoperare caratteri riservati quali l'asterisco (*), il punto interrogativo (?), i due punti (:), la virgola (,), il segno di uguale (=), e così via.

34, SYNTAX ERROR, tt, ss
Mancanza del nome del file. Nell'ultimo comando inviato non è stato inserito il nome del file; oppure il D.O.S. non è in grado di riconosce il nome come tale. Se quest'ultima è la causa, probabilmente si è omesso, nel comando, il simbolo dei due punti; riscrivere correttamente il tutto.

35 / 38
Non esistono messaggi di errore con questo numero.

39, SYNTAX ERROR, tt, ss
Comando non valido. Sul canale dei comandi, avente indirizzo secondario uguale a 15, devono essere inviati solo ed esclusivamente dei comandi. Se la lapallissiana semplicità or ora esposta non ha riscontro pratico di messa in opera, infatti, il 1541 scomoderà il messaggio di errore numero 39 per informarvi del suo infruttuoso tentativo di decifrazione del vostro pseudo comando. Il consiglio è sempre quello di ricontrollare con attenzione i caratteri digitati: "Lui", purtroppo, non sbaglia mai...

40 / 49
Non esistono messaggi di errore con questo numero.

50, RECORD NOT PRESENT, tt, ss
Record non presente. Questo messaggio di errore è principalmente dedicato a chi opera con i file relativi (REL) ed indica che il numero di record richiesto non è ancora stato creato. Nella normale espansione (= aggiunta di record) in un file relativo, sia esso già esistente oppure nuovo, tale errore non avrà luogo come, invece, accade nel caso in cui si posizioni il puntatore in lettura su un record non esistente, oppure si tenti di leggere oltre l'ultimo record esistente.

51, OVERFLOW IN RECORD, tt, ss
Superamento delle dimensioni del singolo record. Sempre dovuto a problemi di gestione dei file relativi, questo messaggio di

errore ha luogo quando i dati che dovrebbero essere scritti nel record corrente (il record è una particolare struttura dati contenente delle indicazioni che ha senso considerare insieme; esempio: nome, cognome e telefono di una persona) superano la dimensione massima del record stesso. A meno di ricreare un nuovo file con record più capienti, l'unico rimedio valido è quello di troncare le informazioni eccedenti, assicurandosi che nel calcolo delle dimensioni del record siano stati inclusi anche i caratteri particolari come, ad esempio, il ritorno carrello CHR$(13) di chiusura della singola informazione.

52, FILE TOO LARGE, tt, ss
File troppo grande. Sempre per i "REL", il messaggio di errore numero 52 informa che sul dischetto non è disponibile lo spazio necessario alla creazione del record relativo richiesto. La spiacevole faccenda può essere generalmente evitata creando l'ultimo numero di record richiesto al momento della creazione dell'intero file. Se, invece, il file è davvero troppo grande per il floppy disk, verificare che non vi siano registrati altri file che non siano di tipo "REL" (al quale tipo dovrebbe sempre essere dedicato un intero dischetto), eventualmente copiandoli altrove, oppure, al limite, suddividere il file su più dischetti od usare dei dati abbreviati onde consentire la riduzione delle dimensioni dei singoli record, aumentando in tal modo lo spazio disponibile sul floppy.

53 / 59
Non esistono messaggi di errore con questo numero.

60, WRITE FILE OPEN, tt, ss
File già aperto in scrittura. Costituisce un'ottima norma di programmazione I/O, l'abitudine di chiudere i file quando non serve più tenerli aperti. Comportandosi altrimenti, infatti, si rischia di fare una OPEN su un file ancora aperto in scrittura. E' corretto chiudere subito un file del genere, per evitare la comparsa dell'asterisco di fianco al tipo di file nella directory, ad indicare un file non chiuso correttamente che, molto probabilmente, dovrà essere definitivamente ucciso da un comando Validate.

61, FILE NOT OPEN, tt, ss
File non aperto. Sostanzialmente equivale a cambiare marcia prima di mettere in moto l'automobile(!); per avere accesso ad un file è assolutamente necessario un preventivo comando di OPEN del file stesso, con l'unica eccezione dei vari LOAD, SAVE, VERIFY, etc. i quali provvedono alla faccenda in maniera completamente automatica: in tutti gli altri casi è il programmatore a gestire l'intera procedura.

62, FILE NOT FOUND, tt, ss
File non trovato sul dischetto correntemente inserito nell'unità a dischi. Rappresenta uno degli errori di "default" quando si inserisce nel drive un dischetto che non è quello giusto. Il D.O.S. informa che il file richiesto con il LOAD, la OPEN, etc., non risulta essere presente sul floppy disk attuale. E' sufficiente riprovare dopo essersi assicurati che il nome del file è scritto correttamente.

63, FILE EXISTS, tt, ss
Nome di file già esistente sul dischetto attuale. Con questo messaggio di errore il D.O.S. informa che sul floppy disk correntemente inserito nel 1541, esiste già un file con lo stesso nome di quello che si vorrebbe scrivere. Dal momento che non sono ammessi più nomi di file uguali nell'ambito della stessa directory, per risolvere il problema sarà sufficiente la scelta di un nominativo differente, oppure l'uso dello stesso nome premesso dai caratteri di "chiocciola" e "due punti" (comunque da evitare) per cancellare e sovrascrivere la vecchia versione su disco con quella nuova da registrare.

64, FILE TYPE MISMATCH, tt, ss
Errore nella specificazione del tipo di file. L'accesso al file richiesto non è possibile con l'uso del tipo di file specificato. L'esempio tipico di generazione è rappresentato dal tentativo di caricare con il comando basic LOAD un file non di tipo "PRG". Onde evitare la comparsa di tale errore è opportuno aver completamente assimilato la parte relativa ai tipi di file, e comportarsi di conseguenza, sapendo a cosa serve ciascun tipo di file.

65, NO BLOCK, tl, sl
Nessun blocco disponibile. Il presente messaggio di errore è essenzialmente "riservato" agli assidui utilizzatori del comando "Block-Allocate" di allocazione dei blocchi; se il settore richiesto risulta essere già occupato, si otterrà in risposta un errore numero 65 nel quale i numeri di traccia e settore "tl" e "sl" indicheranno il primo blocco libero disponibile, a partire dalla traccia 35 a scalare. L'utente ha la facoltà di prendere tale segnalazione come un consiglio (e quindi allocare il settore "sl" della traccia "tl"), oppure di trascurarla in favore di un altro settore a sua scelta. Se il valore di traccia raccomandato è nullo (la traccia numero zero, normalmente, non esiste!) bisogna interpretare la segnalazione come un'indisponibilità del D.O.S. a trovare un altro settore libero: tutti i blocchi rimanenti sono già stati occupati!

66, ILLEGAL TRACK OR SECTOR, tt, ss
Numero di traccia o di settore illegale. Tutti i 683 blocchi di un dischetto (664 liberi + 19 settori da 0 a 18 della traccia riservata alla directory: la diciottesima) sono univocamente individuabili da una coppia di valori rispettivamente indicanti i valori di traccia e di settore per l'identificazione del blocco. Tale coppia di valori può essere tanto quella fornita tra i parametri di un comando di "Block-Read" oppure "U1", quanto il puntatore al blocco successivo presente ai byte 0 ed 1 di ciascun settore. Se una malaugurata alterazione magnetica del supporto (nel caso del puntatore) oppure un valore non corretto come parametro (nel caso dei comandi ad accesso diretto), fornisce al D.O.S. una traccia e/o un settore inesistenti (tipo: traccia 43 e/o settore 21), quest'ultimo si sentirà in dovere di creare nel buffer di uscita il messaggio di errore numero 66, piuttosto di andare a cercare il blocco "spaziale" richiesto. Verificare con un comando di "Validate" se l'errore persiste ancora, se i puntatori dei file sono tutti corretti, oppure, nel caso dell'accesso diretto, controllare i valori di traccia e settore inviati, soprattutto se contenuti in variabili nel contesto di un programma piuttosto grande.

67, ILLEGAL TRACK OR SECTOR, tt, ss
Numero di traccia o di settore di sistema illegale. Il messaggio numero 67 è tra i più desueti dell'intero set degli errori generabili dal D.O.S. del disk drive 1541. Può avere luogo in un file del tipo "REL" nel quale qualche valore dei side-sector ha subito alterazioni più o meno volontarie. L'errore si manifesta allorchè il D.O.S. cerca di attingere dal puntatore relativo al side sector dove si trova il record richiesto e riceve un valore illegale di traccia e/o di settore. L'unica soluzione è quella di impiegare un buon editor per floppy disk del 1541 e rimettere a posto, se possibile, i puntatori ai record alterati.

68 / 69
Non esistono messaggi di errore con questo numero.

70, NO CHANNEL, tt, ss
Il canale richiesto non risulta essere disponibile. Il D.O.S. del disk drive 1541 gestisce tutte le comunicazioni in corso, siano esse di input oppure di output, con la memorizzazione degli indirizzi secondari e relativi canali interni associati in un'apposita tabella la quale, chiaramente, possiede ben precisi limiti di occupazione massima. Precisamente, è possibile la contemporanea apertura di TRE FILE SEQUENZIALI al massimo, oppure UN FILE RELATIVO ED UNO SEQUENZIALE, oltre al canale dei comandi con indirizzo secondario uguale

a 15. Come buona abitudine è opportuno ricordare sempre di inserire, nei comandi, il numero del drive (sebbene sia sempre uguale zero), specialmente nelle OPEN di file sequenziali, per evitare di ritrovarsi con un'indesiderata limitazione massima di gestione contemporanea di soli due file del tipo "SEQ". Altra "legge" da osservare con particolare scrupolo per evitare la comparsa di questo (e di altri!) messaggi di errore, è quella di chiudere immediatamente un file non appena non è più necessario, tenendo aperti nello stesso momento il minor numero di file possibili.

71, DIRECTORY ERROR, tt, ss
Errore nella directory dell'attuale floppy disk. Questo è uno dei tipici errori di chi opera con i comandi ad accesso diretto senza inizializzare i dischetti con "Initialize".

Se durante le varie operazioni I/O il D.O.S. riscontra qualche discordanza tra quanto ricavato dalla B.A.M. e ciò che ha conservato precedentemente dalla stessa (come, per esempio, il numero dei blocchi liberi), ritiene che la directory è "misteriosamente" cambiata ed avverte l'utente con un DIRECTORY ERROR. Il rimedio "istantaneo" è dato dall'invio di un semplice comando "I" per ricaricare in memoria la B.A.M. del nuovo dischetto inserito nel disk drive.

72, DISK FULL, tt, ss
Disco completo. Nella parte relativa alla directory si è visto che a ciascuno dei file contenuti sul dischetto viene associato un "ingresso" presente in traccia 18 in ragione di otto ingressi con le informazioni (tipo di file, nome, traccia e settore di inizio, numero di blocchi occupati sul floppy disk) per altrettanti file, su ciascuno dei settori della diciottesima traccia, a partire dal settore 1, in quanto nel primo blocco (numerato con zero), si trova la Block Availability Map del dischetto. Sapendo che la traccia 18 è costituita in totale da 19 (da 0 a 18) settori, la moltiplicazione 18 settori (da 1 a 18) per 8 ingressi ciascuno fornisce un massimo numero di files ospitabili sul singolo floppy uguale a 144. Raggiunto tale limite, il D.O.S. non saprà più ove memorizzare gli ingressi relativi ad altri file che si vogliono aggiungere e, pertanto, emetterà il messaggio per indicare la saturazione.
L'altro motivo che può dare luogo all'errore numero 72 è il completo riempimento dei "664 Blocks Free" che il 1541 lascia liberi subito dopo la formattazione: se la directory non contiene un numero esagerato di file la causa sarà, molto probabilmente, quest'ultima. Uno dei rimedi possibili è quello di tentare un comando di "Validate" sperando che il D.O.S. riesca a "recuperare" ulteriori settori liberi: nel caso di superamento dei 144 file, se il tentativo risulta essere insufficiente, non resta altro che "spargere" i file molto corti tra più dischetti piuttosto di raggrupparli in uno solo, oppure, se tutti i settori sono già occupati ("0 Blocks Free") si deve semplicemente acquistare qualche altro floppy...

73, CBM DOS V2.6 1541, tt, ss
Versione di D.O.S. del disk drive. Se il canale dei comandi viene controllato subito dopo l'accensione del disk drive 1541 oppure in seguito ad un comando di reset "U:", si otterrà sempre questo messaggio che andrà interpretato come una possibilità di controllo della versione di D.O.S. utilizzata: ad esempio lo stesso messaggio del drive Commodore 1571 a doppia faccia, tipico del C/128, riporta una R.O.M. 1571 con una versione 3.0 del Disk Operating System.

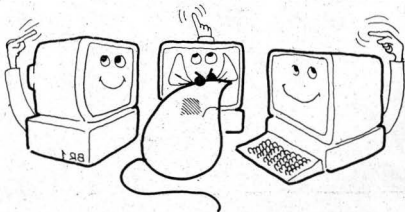
Se fornito in seguito ad un tentativo di scrittura, invece, indica che il formato del dischetto è compatibile solo in lettura con il D.O.S. V2.6 in quanto, probabilmente, è stato creato con differenti versioni come, ad esempio, il sistema operativo della vecchia unità a dischi Commodore 2040.

Molto spesso, però, tale errore indica semplicemente una modifica del byte 2 del settore zero di traccia diciotto (B.A.M.). Mentre nei primi due byte (0 ed 1) sono contenuti i valori rispettivi di 18 ($12) ed 1 ($01) da cui ha inizio l'indice dei file del dischetto, nel byte 2 si trova il valore che il 1541 impiega per stabilire con quale formato ha a che fare e contiene il valore standard di 65 ($41), corrispondente alla lettera "A", visibile anche nell'indicazione "2A" con la quale il D.O.S. contrassegna le directory che crea.

Sostituendo tale riferimento con un valore differente, si trae in inganno l'ignaro D.O.S. V2.6 che si rifiuterà, quindi, di scrivere su un floppy disk che riterrà non suo: questa caratteristica di modifica del byte 2 (il terzo, in realtà, dopo 0 ed 1) del settore zero della diciottesima traccia (l'epica B.A.M...) può, infatti, essere sfruttata per la protezione in scrittura di un dischetto, sul quale non saranno, quindi, più possibili operazioni di "Scratch", SAVE, Validate etc.

74, DRIVE NOT READY, tt, ss
Disk drive non pronto. Questo messaggio di errore indica un tentativo di accesso ad un disk drive che non è "pronto", indicando con tale termine la mancata presenza di un dischetto "valido" nel 1541.

Un esempio di dischetto non valido è un floppy non ancora formattato oppure, più spesso, il mancato inserimento di un qualunque supporto magnetico nell'unità a dischi.

Dal momento che non si tratta di un errore vero e proprio, tale problema può essere risolto semplicemente dando in pasto al drive dei dischetti corretti in tutti i sensi, oppure estraendo e re-inserendo il corrente floppy disk, nel caso di problemi di posizionamento dello stesso.

MAPPA DI MEMORIA

DEL DISK DRIVE COMMODORE 1541

DEC	ESA	DESCRIZIONE
0	$0	Codice comando per buffer 0
1	$1	Codice comando per buffer 1
2	$2	Codice comando per buffer 2
3	$3	Codice comando per buffer 3
4	$4	Codice comando per buffer 4
5	$5	Locazione non usata
6	$6	Traccia per buffer 0
7	$7	Settore per buffer 0
8	$8	Traccia per buffer 1
9	$9	Settore per buffer 1
10	$A	Traccia per buffer 2
11	$B	Settore per buffer 2
12	$C	Traccia per buffer 3
13	$D	Settore per buffer 3
14	$E	Traccia per buffer 4
15	$F	Settore per buffer 4
16	$10	Locazione non usata
17	$11	Locazione non usata
18	$12	ID. 1 per drive 0
19	$13	ID. 2 per drive 0
20	$14	ID. 1 per drive 1
21	$15	ID. 2 per drive 1
22	$16	ID. 1
23	$17	ID. 2
24	$18	N. Traccia dell'intestazione
25	$19	N. Settore dell'intestazione
26	$1A	Parita' per l'intestazione del blocco
27	$1B	Locazione non usata
28	$1C	Flag per protezione in scrittura drive 0
29	$1D	Flag per protezione in scrittura drive 1
30	$1E	Maschera per protezione in scrittura (bit 4 di $1C00)
31	$1F	Locazione non usata
32	$20	Flag per movimento testina drive 0
33	$21	Flag per movimento testina drive 1
34	$22	Traccia attuale
35	$23	Flag per attesa sul bus seriale (comandi UI+ e UI-)
36	$24	Dato della testina di lettura per confronto
37	$25	Dati del disco (intestazione del blocco)
38	$26	Dati del disco (intestazione del blocco)
39	$27	Dati del disco (intestazione del blocco)
40	$28	Dati del disco (intestazione del blocco)
41	$29	Dati del disco (intestazione del blocco)
42	$2A	Dati del disco (intestazione del blocco)
43	$2B	Dati del disco (intestazione del blocco)
44	$2C	Dati del disco (intestazione del blocco)
45	$2D	Locazione non usata
46	$2E	Puntatore low al buffer dati
47	$2F	Puntatore high al buffer dati
48	$30	Puntatore low del buffer per disk controller
49	$31	Puntatore high del buffer per disk controller
50	$32	Puntatore low al buffer attuale
51	$33	Puntatore high al buffer attuale
52	$34	Puntatore low del buffer
53	$35	Puntatore high del buffer
54	$36	Puntatore low nel buffer
55	$37	Puntatore high nel buffer

56	$38	Costante di inizio del blocco dati
57	$39	Costante 8 per inizio dati del blocco di testa
58	$3A	Segnale di parita' per i dati del buffer
59	$3B	Locazione non usata
60	$3C	Byte per stato motore acceso/spento
61	$3D	Numero drive per disk controller
62	$3E	Numero drive attuale
63	$3F	Numero buffer per disk controller
64	$40	Numero di traccia attuale
65	$41	Numero di buffer
66	$42	Offset di traccia nel lavoro attuale
67	$43	Numero di settori per traccia durante la formattazione
68	$44	Codice comando di controllo motore
69	$45	Codice comando di controllo motore
70	$46	Locazione non usata
71	$47	Costante 7 per l'inizio dati del blocco di testa
72	$48	Flag per motore acceso/spento
73	$49	Puntatore di stack durante la routine d'interrupt
74	$4A	Contatore passi per il movimento testina
75	$4B	Contatore di tentativi
76	$4C	Codice comando per confronto
77	$4D	Numero di settore
78	$4E	Puntatore buffer
79	$4F	Puntatore buffer
80	$50	Flag per calcolo parita'
81	$51	Numero di traccia attuale durante la formattazione
82	$52	Byte dati ID. 1
83	$53	Byte dati ID. 2
84	$54	Byte dati
85	$55	Byte dati
86	$56	Puntatore tavola 1 (maschera)
87	$57	Puntatore tavola 2 (maschera)
88	$58	Valore maschera
89	$59	Valore maschera
90	$5A	Valore maschera
91	$5B	Valore maschera
92	$5C	Byte dati
93	$5D	Byte dati
94	$5E	Passo per contatore
95	$5F	Passo per contatore
96	$60	Contatore
97	$61	Contatore
98	$62	Puntatore per routines movimento testina (low)
99	$63	Puntatore per routines movimento testina (high)
100	$64	Passi testina
101	$65	Puntatore a $EB22 per comando UI (low)
102	$66	Puntatore a $EB22 per comando UI (high)
103	$67	Flag di comando
104	$68	Flag per inizializzazione
105	$69	Ampiezza per la divisione di settore
106	$6A	Numero massimo di tentativi di lettura (5)
107	$6B	Puntatore low a $FFEA (tavola indirizzi dei comandi User)
108	$6C	Puntatore high a $FFEA (tavola indirizzi dei comandi User)
109	$6D	Settori liberi nella traccia attuale
110	$6E	Settori liberi nella traccia attuale
111	$6F	Puntatore low indirizzo di JMP per i comandi Memory e Block
112	$70	Puntatore high indirizzo di JMP per i comandi Memory e Block
113	$71	Contatore per l'indirizzo secondario
114	$72	Byte di lavoro per valori decimali
115	$73	Numero di side sector

116	$74	Locazione non usata
117	$75	Indirizzo low di JMP per i comandi User
118	$76	Indirizzo high di JMP per i comandi user
119	$77	Numero di periferica per il LISTEN sommato a 32 ($20)
120	$78	Numero di periferica per il TALK sommato a 64 ($40)
121	$79	Flag per LISTEN attivo
122	$7A	Flag per TALK attivo
123	$7B	Locazione non usata
124	$7C	Flag per ATN di ricezione dati dal bus seriale
125	$7D	Flag per EOI ricevuto dal bus seriale
126	$7E	Ultimo numero di traccia
127	$7F	Numero di drive
128	$80	Numero di traccia
129	$81	Numero di settore
130	$82	Numero del canale
131	$83	Indirizzo secondario
132	$84	Indirizzo secondario
133	$85	Byte di dati per M-R
134	$86	Contatore per il numero di files cancellati
135	$87	Puntatore di scrittura
136	$88	Lunghezza del record
137	$89	Flag di lettura/scrittura
138	$8A	Puntatore al vecchio side sector
139	$8B	Byte di risultato divisione
140	$8C	Byte di lavoro
141	$8D	Byte di lavoro
142	$8E	Byte per il calcolo del numero di blocchi o di side sector
143	$8F	Byte per il calcolo del numero di blocchi o di side sector
144	$90	Puntatore nel blocco dati
145	$91	Numero del record
146	$92	Byte di registro
147	$93	Byte di lavoro
148	$94	Puntatore low al buffer della directory
149	$95	Puntatore high al buffer della directory
150	$96	Locazione non usata
151	$97	Locazione non usata
152	$98	Contatore degli otto bit per la trasmissione seriale
153	$99	Indirizzo low del buffer 0
154	$9A	Indirizzo high del buffer 0
155	$9B	Indirizzo low del buffer 1
156	$9C	Indirizzo high del buffer 1
157	$9D	Indirizzo low del buffer 2
158	$9E	Indirizzo high del buffer 2
159	$9F	Indirizzo low del buffer 3
160	$A0	Indirizzo high del buffer 3
161	$A1	Indirizzo low del buffer 4
162	$A2	Indirizzo high del buffer 4
163	$A3	Indirizzo low del buffer di input
164	$A4	Indirizzo high del buffer di input
165	$A5	Indirizzo low del buffer per i messaggi di errore
166	$A6	Indirizzo high del buffer per i messaggi di errore
167	$A7	Allocazione del buffer per il canale 0
168	$A8	Allocazione del buffer per il canale 1
169	$A9	Allocazione del buffer per il canale 2
170	$AA	Allocazione del buffer per il canale 3
171	$AB	Allocazione del buffer per il canale 4
172	$AC	Allocazione del buffer per il canale 5
173	$AD	Allocazione del buffer per il canale 6
174	$AE	Associazione del buffer per il canale 0
175	$AF	Associazione del buffer per il canale 1

176	$B0	Associazione del buffer per il canale 2
177	$B1	Associazione del buffer per il canale 3
178	$B2	Associazione del buffer per il canale 4
179	$B3	Associazione del buffer per il canale 5
180	$B4	Associazione del buffer per il canale 6
181	$B5	Numero low del record per il canale 0
182	$B6	Numero low del record per il canale 1
183	$B7	Numero low del record per il canale 2
184	$B8	Numero low del record per il canale 3
185	$B9	Numero low del record per il canale 4
186	$BA	Numero low del record per il canale 5
187	$BB	Numero high del record per il canale 0
188	$BC	Numero high del record per il canale 1
189	$BD	Numero high del record per il canale 2
190	$BE	Numero high del record per il canale 3
191	$BF	Numero high del record per il canale 4
192	$C0	Numero high del record per il canale 5
193	$C1	Puntatore in scrittura per i file relativi (canale 0)
194	$C2	Puntatore in scrittura per i file relativi (canale 1)
195	$C3	Puntatore in scrittura per i file relativi (canale 2)
196	$C4	Puntatore in scrittura per i file relativi (canale 3)
197	$C5	Puntatore in scrittura per i file relativi (canale 4)
198	$C6	Puntatore in scrittura per i file relativi (canale 5)
199	$C7	Lunghezza del record per i file relativi (canale 0)
200	$C8	Lunghezza del record per i file relativi (canale 1)
201	$C9	Lunghezza del record per i file relativi (canale 2)
202	$CA	Lunghezza del record per i file relativi (canale 3)
203	$CB	Lunghezza del record per i file relativi (canale 4)
204	$CC	Lunghezza del record per i file relativi (canale 5)
205	$CD	Numero del buffer per i side sector (canale 0)
206	$CE	Numero del buffer per i side sector (canale 1)
207	$CF	Numero del buffer per i side sector (canale 2)
208	$D0	Numero del buffer per i side sector (canale 3)
209	$D1	Numero del buffer per i side sector (canale 4)
210	$D2	Numero del buffer per i side sector (canale 5)
211	$D3	Flag per il comando di input
212	$D4	Puntatore nel record per i file relativi
213	$D5	Numero di blocchi side sector necessari
214	$D6	Puntatore al blocco dati nel side sector
215	$D7	Puntatore nei file relativi
216	$D8	Settore (nella directory) del primo file
217	$D9	Settore (nella directory) del secondo file
218	$DA	Settore (nella directory) del terzo file
219	$DB	Settore (nella directory) del quarto file
220	$DC	Settore (nella directory) del quinto file
221	$DD	Puntatore all'entry della directory per il primo file
222	$DE	Puntatore all'entry della directory per il secondo file
223	$DF	Puntatore all'entry della directory per il terzo file
224	$E0	Puntatore all'entry della directory per il quarto file
225	$E1	Puntatore all'entry della directory per il quinto file
226	$E2	Numero di drive per il primo file
227	$E3	Numero di drive per il secondo file
228	$E4	Numero di drive per il terzo file
229	$E5	Numero di drive per il quarto file
230	$E6	Numero di drive per il quinto file
231	$E7	Tipo del primo file ([illegible], SEQ etc.)
232	$E8	Tipo del secondo file [illegible]PRG, SEQ etc.)
233	$E9	Tipo del terzo file (PRG, SEQ etc.)
234	$EA	Tipo del quarto file (PRG, SEQ etc.)
235	$EB	Tipo del quinto file (PRG, SEQ etc.)

236	$EC	Flag per accesso diretto sul canale 0
237	$ED	Flag per accesso diretto sul canale 1
238	$EE	Flag per accesso diretto sul canale 2
239	$EF	Flag per accesso diretto sul canale 3
240	$F0	Flag per accesso diretto sul canale 4
241	$F1	Flag per accesso diretto sul canale 5
242	$F2	Flag di lettura/scrittura per il canale 0
243	$F3	Flag di lettura/scrittura per il canale 1
244	$F4	Flag di lettura/scrittura per il canale 2
245	$F5	Flag di lettura/scrittura per il canale 3
246	$F6	Flag di lettura/scrittura per il canale 4
247	$F7	Flag di lettura/scrittura per il canale 5
248	$F8	Marker di fine per comando Copy
249	$F9	Numero del buffer/numero del drive
250	$FA	Numero del canale 0
251	$FB	Numero del canale 1
252	$FC	Numero del canale 2
253	$FD	Numero del canale 3
254	$FE	Numero del canale 4
255	$FF	Flag di errore
256	$100	Area di stack del microprocessore
...	$...	Area di stack del microprocessore
442	$1BA	Area di stack del microprocessore
443	$1BB	Bytes di parcheggio letti dalla testina ($1C01)
...	$...	Bytes di parcheggio letti dalla testina ($1C01)
511	$1FF	Bytes di parcheggio letti dalla testina ($1C01)
512	$200	Buffer di input per stringa comando dal bus seriale
...	$...	Buffer di input per stringa comando dal bus seriale
552	$228	Buffer di input per stringa comando dal bus seriale
553	$229	Locazione non usata
554	$22A	Numero della parola di comando
555	$22B	Tabella di associazione del canale per l'indirizzo secondario 0
556	$22C	Tabella di associazione del canale per l'indirizzo secondario 1
557	$22D	Tabella di associazione del canale per l'indirizzo secondario 2
558	$22E	Tabella di associazione del canale per l'indirizzo secondario 3
559	$22F	Tabella di associazione del canale per l'indirizzo secondario 4
560	$230	Tabella di associazione del canale per l'indirizzo secondario 5
561	$231	Tabella di associazione del canale per l'indirizzo secondario 6
562	$232	Tabella di associazione del canale per l'indirizzo secondario 7
563	$233	Tabella di associazione del canale per l'indirizzo secondario 8
564	$234	Tabella di associazione del canale per l'indirizzo secondario 9
565	$235	Tabella di associazione del canale per l'indirizzo secondario 10
566	$236	Tabella di associazione del canale per l'indirizzo secondario 11
567	$237	Tabella di associazione del canale per l'indirizzo secondario 12
568	$238	Tabella di associazione del canale per l'indirizzo secondario 13
569	$239	Tabella di associazione del canale per l'indirizzo secondario 14
570	$23A	Tabella di associazione del canale per l'indirizzo secondario 15
571	$23B	Flag di scrittura per il canale 5
572	$23C	Byte di flag: bit 7 = write flag; bit 6 = read flag; bit 5 = end flag
573	$23D	Byte di flag: bit 7 = write flag; bit 6 = read flag; bit 5 = end flag
574	$23E	Registro di uscita; byte di dati per il canale 0
575	$23F	Registro di uscita; byte di dati per il canale 1
576	$240	Registro di uscita; byte di dati per il canale 2
577	$241	Registro di uscita; byte di dati per il canale 3
578	$242	Registro di uscita; byte di dati per il canale 4
579	$243	Registro di uscita; byte di dati per il canale 5
580	$244	Puntatore di fine per il canale 0
581	$245	Puntatore di fine per il canale 1

```
582     $246    Puntatore di fine per il canale 2
583     $247    Puntatore di fine per il canale 3
584     $248    Puntatore di fine per il canale 4
585     $249    Puntatore di fine per il canale 5
586     $24A    Tipo di file: 0=DEL; 1=SEQ; 2=PRG; 3=USR; 4=REL
587     $24B    Lunghezza massima del nome
588     $24C    Indirizzo secondario
589     $24D    Codice per lettura ($80) o scrittura ($90)
590     $24E    Massimo numero di settori nella traccia
591     $24F    Flag per comando Copy
592     $250    Flag per la presenza del canale
593     $251    Flag di alterazione B.A.M. per il drive 0
594     $252    Flag di alterazione B.A.M. per il drive 1
595     $253    Flag di indicazione dell'entry della directory cercato
596     $254    Flag per la directory
597     $255    Flag per comando ($0=nessuno)
598     $256    Registro di allocazione del canale per il comando Copy
599     $257    Numero del canale
600     $258    Lunghezza del record
601     $259    Traccia del side sector
602     $25A    Settore del side sector
603     $25B    Lunghezza del record o codice di comando per il buffer 0
604     $25C    Lunghezza del record o codice di comando per il buffer 1
605     $25D    Lunghezza del record o codice di comando per il buffer 2
606     $25E    Lunghezza del record o codice di comando per il buffer 3
607     $25F    Lunghezza del record o codice di comando per il buffer 4
608     $260    Puntatore al blocco della directory per il canale 0
609     $261    Puntatore al blocco della directory per il canale 1
610     $262    Puntatore al blocco della directory per il canale 2
611     $263    Puntatore al blocco della directory per il canale 3
612     $264    Puntatore al blocco della directory per il canale 4
613     $265    Puntatore al blocco della directory per il canale 5
614     $266    Puntatore nella directory per il canale 0
615     $267    Puntatore nella directory per il canale 1
616     $268    Puntatore nella directory per il canale 2
617     $269    Puntatore nella directory per il canale 3
618     $26A    Puntatore nella directory per il canale 4
619     $26B    Puntatore nella directory per il canale 5
620     $26C    Flag per errori (0=nessuno)
621     $26D    Maschera per il Led del drive
622     $26E    Ultimo numero di drive
623     $26F    Ultimo numero di settore
624     $270    Numero del canale
625     $271    Locazione non usata
626     $272    Numero di blocchi low per i file relativi
627     $273    Numero di drive per l'intestazione/numero di blocchi high per i
file relativi
628     $274    Lunghezza della linea di comando nel buffer di input
629     $275    Primo carattere del buffer di input/carattere di ricerca
630     $276    Fine del nome nel comando
631     $277    Contatore di parametro (virgola etc.)
632     $278    Numero di nomi di file nel comando
633     $279    Flag per il controllo della presenza di un file
634     $27A    Puntatore alla linea di comando
635     $27B    Posizione della virgola
636     $27C    Byte di ricerca
637     $27D    Byte di ricerca
638     $27E    Byte di ricerca
639     $27F    Locazione non usata
640     $280    Numero di traccia per il primo file nella directory
```

```
641     $281    Numero di traccia per il secondo file nella directory
642     $282    Numero di traccia per il terzo file nella directory
643     $283    Numero di traccia per il quarto file nella directory
644     $284    Numero di traccia per il quinto file nella directory
645     $285    Numero di settore per il primo file nella directory
646     $286    Numero di settore per il secondo file nella directory
647     $287    Numero di settore per il terzo file nella directory
648     $288    Numero di settore per il quarto file nella directory
649     $289    Numero di settore per il quinto file nella directory
650     $28A    Flag "wildcard"
651     $28B    Flag per il controllo della sintassi
652     $28C    Flag di sintassi per la ricerca del drive
653     $28D    Puntatore per la directory
654     $28E    Ultimo numero di drive
655     $28F    Flag per la presenza di ulteriori files
656     $290    Numero di settore
657     $291    Flag per la lettura di un blocco dalla directory
658     $292    Puntatore al buffer nella directory
659     $293    Numero di traccia
660     $294    Puntatore al buffer della directory
661     $295    Contatore degli ingressi (entries) della directory
662     $296    Flag per il comando di input/tipo di file (1/4)
663     $297    Modo di controllo operazioni: 0=LOAD, 1=SAVE, 2=APPEND,
4=operazione su file REL, etc.
664     $298    Flag di errore
665     $299    Contatore per la ricerca di una traccia
666     $29A    Numero di tentativi di lettura per la ricerca di una traccia
667     $29B    Numero di drive 1 per la B.A.M.
668     $29C    Numero di drive 1 per la B.A.M.
669     $29D    Traccia per il drive 0
670     $29E    Traccia per il drive 1
671     $29F    Locazione non usata
672     $2A0    Locazione non usata
673     $2A1    Puntatore low per il buffer 0
674     $2A2    Puntatore high per il buffer 0
675     $2A3    Puntatore low per il buffer 1
676     $2A4    Puntatore high per il buffer 1
677     $2A5    Puntatore low per il buffer 2
678     $2A6    Puntatore high per il buffer 2
679     $2A7    Puntatore low per il buffer 3
680     $2A8    Puntatore high per il buffer 3
681     $2A9    Puntatore low per il buffer 4
682     $2AA    Puntatore high per il buffer 4
683     $2AB    Locazione non usata
...     $...    Locazione non usata
688     $2B0    Locazione non usata
Nota: da 689 ($2B1) a 724 ($2D4) si trova il buffer della directory.
689     $2B1    Valore #$12 (RVSON)
690     $2B2    Valore #$22 (virgolette)
691     $2B3    Primo carattere del nome del disco/nome del file
...     $...    Nome del disco/nome del file (caratteri 2/15)
706     $2C2    Ultimo carattere del nome del disco/nome del file
707     $2C3    Valore #$22 (virgolette)
708     $2C4    Eventuale valore #$2A (asterisco) per un file non chiuso
correttamente
709     $2C5    Prima lettera del tipo di file (esempio: P)
710     $2C6    Seconda lettera del tipo di file (esempio: R)
711     $2C7    Terza lettera del tipo di file (esempio: G)
712     $2C8    Eventuale valore #$3C (segno di minore) per un file protetto dalla
cancellazione
```

```
713     $2C9   Seguito del buffer della directory
...     $...   Seguito del buffer della directory
724     $2D4   Seguito del buffer della directory
Nota: da 725 ($2D5) a 761 ($2F9) si trova il buffer per i messaggi di errore.

725     $2D5   Primo carattere dello stato disco
...     $...   Caratteri successivi dello stato disco
760     $2F8   Ultimo (eventuale) carattere dello stato disco
761     $2F9   Ultima locazione del buffer per i messaggi di errore: Flag di
errore/somma di controllo (checksum)
762     $2FA   Numero low di blocchi liberi per il drive 0
763     $2FB   Numero low di blocchi liberi per il drive 1
764     $2FC   Numero high di blocchi liberi per il drive 0
765     $2FD   Numero high di blocchi liberi per il drive 1
766     $2FE   Messaggio di ritorno dal disk controller per il drive 0
767     $2FF   Messaggio di ritorno dal disk controller per il drive 1
768     $300   R.A.M. libera per il buffer numero 0
...     $...   R.A.M. libera per il buffer numero 0
1023    $3FF   R.A.M. libera per il buffer numero 0
1024    $400   R.A.M. libera per il buffer numero 1
...     $...   R.A.M. libera per il buffer numero 1
1279    $4FF   R.A.M. libera per il buffer numero 1
1280    $500   R.A.M. libera per il buffer numero 2
...     $...   R.A.M. libera per il buffer numero 2
1535    $5FF   R.A.M. libera per il buffer numero 2
1536    $600   R.A.M. libera per il buffer numero 3
...     $...   R.A.M. libera per il buffer numero 3
1791    $6FF   R.A.M. libera per il buffer numero 3
1792    $700   R.A.M. libera per il buffer numero 4
...     $...   R.A.M. libera per il buffer numero 4
2047    $7FF   R.A.M. libera per il buffer numero 4

2048    $800   Memoria non implementata
....    $...   Memoria non implementata
6143    $17FF  Memoria non implementata

Nota: da 6144 ($1800) a 6158 ($180E) si trova il primo Versatile Interface
Adapter 6522, comprendente la porta per il bus seriale:

6144    $1800  Porta B IEEE; registro dati:
Bit 0   Data byte IN (0=Si; 1=No)
Bit 1   Data byte OUT (0=Low; 1=High)
Bit 2   Clock IN (0=Si; 1=No)
Bit 3   Clock OUT (0=Low; 1=High)
Bit 4   Linea dati ATN A (0=Output; 1=Input)
Bit 5   Numero di periferica (Vedere il bit seguente)
Bit 6   Numero di periferica (00=8; 01=9; 10=10; 11=11)
Bit 7   E.O.I. per l'Handshake (0=Si; 1=No)

6145    $1801  Porta A IEEE; registro dati
6146    $1802  Registro direzione dati per la porta B (0=IN; 1=OUT)
6147    $1803  Registro direzione dati per la porta A (1=OUT; 0=IN)
6148    $1804  Flag di interrupt del Timer 1
6149    $1805  Timer 1 High (Bit 7 = 0: fine corsa)
6150    $1806  Numero dei cicli di attesa (1 millisecondo = 98 (#$62) cicli)
6151    $1807  Timer 1 Low
6152    $1808  Registro non utilizzato (Timer a corsa libera)
6153    $1809  Registro non utilizzato (Timer a corsa libera)
6154    $180A  Registro non utilizzato (Non documentato)
6155    $180B  Controllo Timer 1; se il bit 6 = 1 allora il Timer procede a corsa
libera
```

```
6156   $180C  Flag CA1 (ATN IN) per trigger ON
6157   $180D  Registro di flag degli interrupt: bit 1=Bus seriale; bit 6=Timer 1
(0=IRQ non attivo; 1=IRQ attivo)
6158   $180E  Flag di interrupt attraverso ATN IN

6159   $180F  Registro non utilizzato nel 1541 originale. Solo nel 1571, il Bit
5 segnala la modalita' operativa corrente: 0=1541; 1=1571.

6160   $1810  Memoria non implementata
....   $...   Memoria non implementata
7167   $1BFF  Memoria non implementata

Nota: da 7168 ($1C00) a 7182 ($1C0E) si trova il secondo Versatile Interface
Adapter 6522, comprendente la porta di controllo del motore e della testina di
lettura/scrittura:

7168   $1C00  Porta B IEEE; controllo di porta:
Bit 0  Motore 1 per il movimento della testina (0=On; 1=Off)
Bit 1  Motore 0 per il movimento della testina (0=On; 1=Off)
Bit 2  Pilotaggio del motore (0=Off; 1=On)
Bit 3  Accensione del Led di lettura/scrittura (0=Off; 1=On)
Bit 4  Flag di protezione in scrittura (0=Tacca di protezione presente; 1=Tacca
di protezione assente)
Bit 5  Bit 1 di controllo per il motore (0=On; 1=Off)
Bit 6  Bit 2 di controllo per il motore (0=On; 1=Off)
Bit 7  Flag di controllo della presenza del sincronismo (0=Sync #%01010101
(#$55); 1=Sync #%10101010 (#$AA))

7169   $1C01  Porta A IEEE; dati da e per la testina di lettura/scrittura
7170   $1C02  Registro direzione dati per la porta B (0=IN; 1=OUT)
7171   $1C03  Registro direzione dati per la porta A (0=IN/Read; 1=OUT/Write)
7172   $1C04  Flag di interrupt del Timer 1
7173   $1C05  Timer 1 High
7174   $1C06  Timer 1 low (latch)
7175   $1C07  Timer 1 high (latch)
7176   $1C08  Registro non utilizzato (Timer a corsa libera)
7177   $1C09  Registro non utilizzato (Timer a corsa libera)
7178   $1C0A  Registro non utilizzato (Non documentato)
7179   $1C0B  Controllo Timer 1; se i bits 5 & 6 = 1 allora il Timer procede a
corsa libera
7180   $1C0C  Registro di controllo PCR Read/Write
7181   $1C0D  Registro di flag degli interrupt: bit 1=Bus seriale; bit 6=Timer 1
(0=IRQ non attivo; 1=IRQ attivo)
7182   $1C0E  Flag degli interrupts abilitati (#$7F=Interrupts Off)

7184   $1C10  Memoria non implementata
....   $...   Memoria non implementata
49407  $C0FF  Memoria non implementata

49408  $C100  Inizio della R.O.M. del D.O.S. V2.6

Indirizzi di inizio delle routines principali:
51335  $C823  Routine del comando "S" (Scratch)
51393  $C8C1  Routine del comando "D" (Backup); risponde con l'errore 31 di
Syntax Error in quanto il comando, pur presente nella tavola comandi, non ha una
routine R.O.M. che lo esegua.
51440  $C8F0  Routine del comando "C" (Copy)
51848  $CA88  Routine del comando "R" (Rename)
51960  $CAF8  Routine di analisi dei comandi "M" (Memory-Read/Write/Execute)
52000  $CB20  Routine del comando "M-R" (Memory-Read)
52048  $CB50  Routine del comando "M-W" (Memory-Write)
```

```
52060  $CB5C  Routine di analisi dei comandi "U" (User)
52100  $CB84  Routine di apertura del canale ad accesso diretto (#)
52251  $CC1B  Routine di analisi dei comandi "B"
(Block-Allocate/Free/Read/Write/Execute/Pointer)
52335  $CC6F  Routine di accettazione parametri per i comandi "B"
52469  $CCF5  Routine del comando "B-F" (Block-Free)
52483  $CD03  Routine del comando "B-A" (Block-Allocate)
52566  $CD56  Routine del comando "B-R" (Block-Read)
52575  $CD5F  Routine del comando "U1" (simile a Block-Read)
52595  $CD73  Routine del comando "B-W" (Block-Write)
52631  $CD97  Routine del comando "U2" (simile a Block-Write)
52643  $CDA3  Routine del comando "B-E" (Block-Execute)
52669  $CDBD  Routine del comando "B-P" (Block-Pointer)
53253  $D005  Routine del comando "I" (Initialize)
53314  $D042  Routine di caricamento in memoria della Block Availability Map
(B.A.M.)
53365  $D075  Routine di calcolo del numero totale di blocchi liberi (Blocks
free) del dischetto
55220  $D7B4  Routine di apertura per un comando di OPEN con indirizzo
secondario diverso da 15 (canale dei comandi)
55541  $D8F5  Routine di riscrittura di un file (comando "@")
55893  $DA55  Routine di apertura della directory ($)
56000  $DAC0  Routine di chiusura (comando CLOSE)
57042  $DED2  Routines di trattamento Side sectors & files relativi
57863  $E207  Routine del comando "P" (Record)
58620  $E4FC  Tavola dei messaggi di errore
59264  $E780  Routine di controllo per l'Auto-start (Caricamento ed esecuzione
del primo &-File su disco); nelle successive versioni della R.O.M. del 1541 tale
routine non e' piu' stata inclusa, sostituita da un RTS (op. code #$60) in $E780
59299  $E7A3  Routine del comando "&" (Files USR in linguaggio macchina,
eseguibili nella R.A.M. del 1541 con un comando di OPEN)
60014  $EA6E  Routine diagnostica di lampeggiamento del Led per difetti di tipo
Hardware
60064  $EAA0  Routine di reset
60194  $EB22  Routine del comando "UI" (reset parziale)
60574  $EC9E  Routine di caricamento della directory
60804  $ED84  Routine del comando "V" (Validate)
60941  $EE0D  Routine del comando "N" (Header)
61111  $EEB7  Routine di creazione della B.A.M.
62128  $F2B0  Routine di gestione (in interrupt) del disk controller: comandi
JOB QUE etc.
62736  $F510  Routine di lettura dell'intestazione del blocco
64199  $FAC7  Routine di formattazione
65127  $FE67  Routine di interrupt
65281  $FF01  Routine del comando "UI+" o "UI-"
65296  $FF10  Serie di locazioni non usate; nelle successive versioni del 1541,
le correzioni e le modifiche sono state poste in quest'area e collegate al resto
del D.O.S. con dei semplici JMP
65514  $FFEA  Tabella standard dei vettori utente per i comandi "U"
65532  $FFFC  Vettore low della routine di reset
65533  $FFFD  Vettore high della routine di reset
65534  $FFFE  Vettore low della routine d'interrupt
65535  $FFFF  Vettore high della routine d'interrupt
```

```
1          LST   OFF
2
3 *********************************
4 *  SPEEDISK-LOAD VERSIONE 2.1   *
5 *********************************
6
7 *********************************
8 * TURBO-DISK PER IL DISK DRIVE *
9 * 1541 ED IL COMMODORE 64.     *
10 ********************************
11
12 ********************************
13 * VERSIONE PER "SPECIALE 1541" *
14 * COPYRIGHT (C) '88 BY SYSTEMS *
15 ********************************
16
17 ********************************
18 *  CARATTERISTICHE PRINCIPALI  *
19 *------------------------------*
20 * 1. UTILIZZO DELLO STACK PER  *
21 *    LA MEMORIZZAZIONE DEI     *
22 *    VALORI DI LAVORO PER NON  *
23 *    ALTERARE LA PAGINA ZERO.  *
24 *..............................*
25 * 2. STORAGGIO E VERIFICA SU   *
26 *    TUTTA LA R.A.M. DEL C/64. *
27 *..............................*
28 * 3. ABILITAZIONE DELL'I/O CON *
29 *    LA CASSETTA E DEL LOAD    *
30 *    DELLA DIRECTORY.          *
31 *..............................*
32 * 4. POSSIBILITA' DI ABORTIRE  *
33 *    IL CARICAMENTO CON IL     *
34 *    TASTO DI RUN/STOP.        *
35 *..............................*
36 * 5. IMPLEMENTAZIONE TOTALE    *
37 *    SUL COMMODORE 64 A PROVA  *
38 *    DI RUN/STOP & RESTORE.    *
39 *..............................*
40 * 6. POSSIBILITA' DI ABILITARE *
41 *    O DISABILITARE IL TURBO   *
42 *    CON IL CARATTERE "←".     *
43 *..............................*
44 * 7. VISUALIZZAZIONE IN TEMPO  *
45 *    REALE DEL NUMERO DEI      *
46 *    BLOCCHI CARICATI.         *
47 *..............................*
48 * 8. VISUALIZZAZIONE DEGLI     *
49 *    INDIRIZZI DI CARICAMENTO  *
50 *    INIZIALE E FINALE.        *
51 ********************************
52
53 ********************************
54 *      ETICHETTE ESTERNE.      *
55 ********************************
56
57 * DISABILITA R.A.M., R.O.M. & I/O
58 R6510     =     $01
59
60 * VARIABILE DI STATO I/O
61 ST        =     $90
62
63 * FLAG 0-LOAD 1-VERIFY
64 VERCK     =     $93
65
66 * TMP PER X DURANTE CHRIN
67 TEMPX     =     $97
68
69 * USATI PER RICHIEDERE AL DRIVE
70 * IL PROSSIMO DOPPIO BIT E PER
71 * MANDARE IL TURBO AL DRIVE
72 ZA4       =     $A4
73 ZA5       =     $A5
74
75 * INDIRIZZO DI FINE PROGRAMMA
76 * ALLA FINE DEL TURBO
77 ZAE       =     $AE
78
79 * N. CARATTERI NOME PROGRAMMA
80 FNLEN     =     $B7
81
82 * N. ATTUALE FILE LOGICO
83 LA        =     $B8
84
85 * ATTUALE INDIRIZZO SECONDARIO
86 SA        =     $B9
87
88 * ATTUALE N. PERIFERICA
89 FA        =     $BA
90
91 * PUNTATORE NOME PROGRAMMA
92 FNADR     =     $BB
93
94 * INDIRIZZO DI CARICAMENTO ,8,0
95 USER      =     $C3
96
97 * PRIMO PUNTATORE DI TRANSFER
98 PT1       =     $FB
99
100 * SECONDO PUNTATORE DI TRANSFER
101 PT2       =     $FD
102
103 * VETTORE PER I COMANDI BASIC
104 IGONE     =     $0308
105
106 * VETTORE NMI DEL TASTO RESTORE
107 NMINV     =     $0318
108
109 * VETTORE PER ROUTINE LOAD
110 ILOAD     =     $0330
111
112 * INDIRIZZO VIDEO CONTABLOCCHI
113 CBLOCK    =     $0425
114
115 ********************************
116 *     ETICHETTE PER L'I/O.     *
117 ********************************
118
119 * VIC II --------------------- *
120
```

```
121 * INDICA LA POSIZIONE DEL RASTER
122 RASLIN   -     $D012
123
124 * REGISTRO DI ABILITAZIONE SPRITE
125 ENASPR   -     $D015
126
127 * CIA 1 ---------------------- *
128
129 * SCRITTURA COLONNE TASTIERA
130 PRA1     -     $DC00
131
132 * LETTURA RIGHE TASTIERA
133 PRB1     -     $DC01
134
135 * CIA 2 ---------------------- *
136
137 * CONTROLLA BUS, RS232, VIC ADDR.
138 BUS      -     $DD00
139
140 **********************************
141 *     ROUTINES DEL KERNAL.       *
142 **********************************
143
144 * INVIA UN BYTE ALLO SCHERMO
145 CHROUT   -     $E716
146
147 * INVIA "TALK" SUL BUS SERIALE
148 TALK     -     $ED09
149
150 * INVIA "LISTEN" SUL BUS SERIALE
151 LISTEN   -     $ED0C
152
153 * INDIRIZZO SECONDARIO PER LISTEN
154 SALIST   -     $EDB9
155
156 * INDIRIZZO SECONDARIO PER TALK
157 SATALK   -     $EDC7
158
159 * INVIA UN BYTE SUL BUS SERIALE
160 OUTBYT   -     $EDDD
161
162 * INVIA "UNTALK" SUL SERIAL BUS
163 UNTALK   -     $EDEF
164
165 * INVIA "UNLISTEN" SUL SERIAL BUS
166 ULSTEN   =     $EDFE
167
168 * RICEVE UN BYTE DAL BUS SERIALE
169 INBYTE   -     $EE13
170
171 * RIPRISTINA LE PERIFERICHE I/O
172 RSTIO    -     $F333
173
174 * APERTURA SUL BUS SERIALE
175 BOPEN    -     $F3D5
176
177 * SCRIVE "SEARCHING FOR"
178 PRLOAD   -     $F5AF
179
180 * SCRIVE "LOADING"/"VERIFYING"
181 LDVER    -     $F5D2
182
183 * CHIUSURA DEL BUS SERIALE
184 BCLOSE   -     $F646
185
186 * ERRORE: "?FILE NOT FOUND"
187 FERROR   -     $F704
188
189 **********************************
190 * ETICHETTE PER IL DISK DRIVE. *
191 **********************************
192
193 * NUMERO DEI TENTATIVI DI LETTURA
194 * (USA LA TRACCIA PER IL BUFFER
195 *  NUMERO 3 COME "DEPOSITO")
196 MAXTENT  =     $0C
197
198 * TRACCIA PER IL BUFFER 4 ($700)
199 TKBFR4   -     $0E
200
201 * SETTORE PER IL BUFFER 4 ($700)
202 SKBFR4   -     $0F
203
204 * TRACCIA DELL'INTESTAZIONE FILE
205 TKHEADER -     $18
206
207 * SETTORE DELL'INTESTAZIONE FILE
208 SKHEADER -     $19
209
210 * PUNTATORE AL BUFFER PER IL 1541
211 PTR      -     $30
212
213 * INIZIO STACK PER LA C.P.U. 6502
214 STACK    -     $100
215
216 * BUS SERIALE DEL DRIVE: PORTA B
217 BUSDRIVE -     $1800
218
219 * TESTINA R/W DEL DRIVE: PORTA A
220 TESTINA  =     $1C01
221
222 * TIMER 1 DEL DRIVE: VALORE HIGH
223 TIMER1   -     $1C05
224
225 **********************************
226 *            INIZIO.             *
227 **********************************
228
229 **********************************
230 * INTESTAZIONE BASIC PER IL      *
231 * POSIZIONAMENTO IN MEMORIA      *
232 * DELLO SPEEDISK.                *
233 **********************************
234
235           ORG   $801
236
237           DA    TWOBRK      ; LINK
238           DA    1988        ; N.LINEA
239           DFB   $9E         ; "SYS"
240           TXT   '2061'
```

```
241             BRK                    ; FINE DELLA LINEA BASIC
242 TWOBRK      DA      0              ; FINE DEL PROGRAMMA
243
244 * CONSERVA IL VECCHIO INDIRIZZO DEL LOAD
245             LDX     ILOAD
246             LDY     ILOAD+1
247             STX     $7FE
248             STY     $7FF
249
250 * INIZIALIZZA LO STACK PER EVITARE SOVRAPPOSIZIONI
251             LDX     #$FF
252             TXS
253
254 * PREPARA I PUNTATORI PER POSIZIONARE LO SPEEDISK DA $A000
255             LDX     #<ORIGIN
256             LDY     #>ORIGIN
257             STX     PT1
258             STY     PT1+1
259             LDX     #<DATA
260             LDY     #>DATA
261             STX     PT2
262             STY     PT2+1
263
264 * SETTA I COLORI DI SFONDO E BORDO
265             LDA     #$0B
266             LDY     #$00
267             STA     $D020
268             STY     $D021
269
270 * TRASFERIMENTO DEI BYTES
271 TR1         LDA     (PT2),Y
272             STA     (PT1),Y
273             INY
274             BNE     TR1
275             INC     PT1+1
276             INC     PT2+1
277             LDA     PT2+1
278             CMP     #>FINISH-ORIGIN+DATA
279             BCC     TR1
280
281 * TRASFERIMENTO NELLO STACK DA $10A
282             LDX     #FSTACK-ISTACK
283 TR2         LDA     DATA+FINISH-ORIGIN-1,X
284             STA     STACK+$A-1,X
285             DEX
286             BNE     TR2
287
288 * TRASFERIMENTO NELLA MEMORIA DI FINE SCHERMO DA $7E8
289             LDX     #FSCR-ISCR
290 TR3         LDA     DATA+FINISH-ORIGIN+FSTACK-ISTACK-1,X
291             STA     $7E8-1,X
292             DEX
293             BNE     TR3
294
295 * INIZIALIZZA L'INTERPRETE BASIC
296             JSR     $E3BF
297
298 * SETTA IL COLORE DEI CARATTERI
299             LDA     #$97
300             JSR     CHROUT
```

```
301
302 * STAMPA IL MESSAGGIO DI APERTURA COMMODORE 64 BASIC V2.0 ETC.
303          JSR    $E422
304
305 * STAMPA IL MESSAGGIO "SPEEDISK V2.1 ON ..." ETC.
306          LDA    #<SDMSG
307          LDY    #>SDMSG
308          JSR    $AB1E
309
310 * ABILITA LO SPEEDISK
311          JSR    ENABLE
312
313 * MODIFICA IL PUNTATORE "IGONE" PER IL COMANDO "←"
314          LDX    #<EXESTA
315          LDY    #>EXESTA
316          STX    IGONE
317          STY    IGONE+1
318
319 * VETTORE DI N.M.I.: SPEEDISK IMMUNE AL RESTORE
320          LDX    #<NEWNMI
321          LDY    #>NEWNMI
322          STX    NMINV
323          STY    NMINV+1
324
325 * RE-INIZIALIZZA LO STACK POINTER
326          LDX    #$FF
327          TXS
328
329 * STAMPA IL "READY." ED ATTENDE EVENTUALI COMANDI
330          JMP    $A474
331
332 ********************************
333 * MESSAGGIO DI APERTURA.       *
334 ********************************
335
336 SDMSG    HEX    0D,09,8E,98
337          TXT    ' speedisk v2.1 on - (@) 1988 by systems',0D,98,00
338
339 *------------------------------*
340
341 DATA
342
343          ORG    $A000
344
345 ********************************
346 * TURBO CHE VA AL 1541 DA $700 *
347 * IN POI SOTTO FORMA DI BYTES. *
348 * SYS DI PARTENZA = $07AB.     *
349 ********************************
350
351 * BUFFER $600 PER STORAGGIO DATI
352 ORIGIN   LDA    #$06
353          STA    PTR+1
354
355 * TROVA L'INIZIO DEL BLOCCO DATI
356 LP0      JSR    $F50A
357
358 * LEGGE UN BLOCCO DAL DISCHETTO
359 LP1      BVC    LP1
360          CLV
```

```
361          LDA    TESTINA
362          STA    (PTR),Y
363          INY
364          BNE    LP1
365
366 * LEGGE INTESTAZIONI DEL BLOCCO
367          LDY    #$BA
368 LP2      BVC    LP2
369          CLV
370          LDA    TESTINA
371          STA    STACK,Y
372          INY
373          BNE    LP2
374
375 * CALCOLO CHECKSUM INTESTAZIONE
376          JSR    $F8E0
377
378 * CONTROLLO INTESTAZIONE BLOCCO
379          LDA    $38          ;COSTANTE
380          CMP    $47          ; = 7?
381          BEQ    LP3          ; SI, OK!
382
383 * 22, READ ERROR,Traccia,Settore
384          LDA #$04
385          BNE MSG
386
387 * CALCOLO CHECKSUM DEL BLOCCO
388 LP3      JSR    $F5E9
389          CMP    $3A          ; = CHK2?
390          BEQ    LP4          ; SI, OK!
391
392 * 23, READ ERROR,Traccia,Settore
393          LDA #$05
394          BNE MSG
395
396 * LEGGE LA TRACCIA DEL PROSSIMO
397 * EVENTUALE SETTORE DA CARICARE
398 LP4      LDA    (PTR),Y
399
400 * PROSEGUE SE LA TRACCIA NON E' 0
401 * SE, CIOE', CI SONO ULTERIORI
402 * BLOCCHI DA CARICARE.
403          BNE    LP5
404
405 * ALTRIMENTI INCREMENTA IL NUMERO
406 * DI BYTES DA CARICARE NEL BLOCCO
407 * PRESENTE.
408          INC    $601
409
410 *********************************
411 * INVIA 256 BYTES AL COMPUTER. *
412 *********************************
413
414 LP5      LDA    (PTR),Y
415 ROUTINE  TAX
416
417 * ATTENDE PER L'HANDSHAKE
418 LP6      BIT    BUSDRIVE
419          BPL    LP6
420
421 * MODIFICA LA LINEA "ATN" DEL BUS
422          LDA    #$10
423          STA    BUSDRIVE
424
425 * ATTENDE L'E.O.I. DAL COMPUTER
426 LP7      BIT    BUSDRIVE
427          BMI    LP7
428
429 *********************************
430 * INVIA IL SINGOLO BYTE POSTO  *
431 * NEL REGISTRO X.              *
432 *********************************
433
434          TXA
435
436 * PONE IL NIBBLE "HIGH" IN "LOW"
437          LSR
438          LSR
439          LSR
440          LSR
441
442 * INVIA IL BYTE AL BUS SERIALE
443          STA    BUSDRIVE
444          ASL
445          AND    #$0F
446          STA    BUSDRIVE
447
448 * INVIA IL NIBBLE "LOW" AL BUS
449          TXA
450          AND    #$0F
451          STA    BUSDRIVE
452          ASL
453          AND    #$0F
454          STA    BUSDRIVE
455
456 * FINE INVIO BYTE: PONE "HIGH"
457 * LA LINEA "DATA OUT"
458          LDX    #$0F
459          NOP
460          STX    BUSDRIVE
461
462 * VERIFICA SE ULTIMO SETTORE
463 LAST     LDA    $600         ; TRACCIA
464          BEQ    LP8          ; = 0?
465
466 * INVIA IL PROSSIMO BYTE
467          INY
468          BNE    LP5
469
470 *********************************
471 *   FINE DEL SETTORE ATTUALE.   *
472 *********************************
473
474 * RIPRISTINA IL NUMERO DEI
475 * TENTATIVI DI LETTURA
476          LDA    #$05
477          STA    MAXTENT
478
479 * SETTA IL PROSSIMO NUMERO DI
480 * SETTORE DA LEGGERE
```

```
481            LDA    $601
482            STA    SKBFR4
483
484 * SE LA TRACCIA E' LA STESSA DI
485 * QUELLA ATTUALE ALLORA NON ESCE
486 * E LEGGE IL SUCCESSIVO SETTORE
487            LDA    (PTR),Y
488            CMP    TKBFR4
489            STA    TKBFR4
490            BEQ    LP0JMP
491
492 * ALTRIMENTI ESCE SENZA ERRORI
493            LDA #$01
494
495 * ROUTINE DI GESTIONE ERRORI
496 MSG        JMP $F969
497
498 * LEGGE ALTRI SETTORI NELLA
499 * TRACCIA PRESENTE
500 LP0JMP     JMP    LP0-ORIGIN+$700
501
502 * LEGGE TUTTI I BYTES DELL'ULTIMO
503 * DEI SETTORI DEL FILE RICHIESTO
504 LP8        INY
505            CPY    $601          ; = MAX?
506            BNE    LP5
507
508 * SE Y = PEEK($601) (MAX N. DI
509 * BYTES NEL SETTORE ATTUALE)
510 * ALLORA ESCE CON UN ERRORE
511 * "FITTIZIO" (#$7F) COME "FLAG"
512 * DI FINE DEL FILE
513            LDA #$7F
514            BNE MSG
515
516 * BYTES RIEMPITIVI DEL BUFFER
517            HEX    00,00,00,00,00
518            HEX    00,00,00,00,00
519            HEX 00,00,00,00,00
520
521 *********************************
522 * INIZIO DELLA ROUTINE CON M-E *
523 *********************************
524
525 * SETTA IL VALORE DEL TIMER 1
526 EXEC       LDA    #$10
527            STA    TIMER1
528
529 * SETTA I VALORI INIZIALI DI
530 * TRACCIA & SETTORE A QUELLI
531 * DELL'INTESTAZIONE DEL FILE
532 * RICHIESTO
533            LDA    TKHEADER
534            STA    TKBFR4
535            LDA    SKHEADER      ;BFR N.4
536            STA    SKBFR4
537
538 * SETTA A CINQUE IL NUMERO
539 * MASSIMO DI TENTATIVI DI LETTURA
540 LP9        LDA    #$05
541            STA    MAXTENT
542
543 * ESEGUE CON IL COMANDO EXECUTE
544 * #$E0 LA ROUTINE DEL BUFFER N.
545 * 4 POSTA A PARTIRE DA $700
546 LP10       LDA    #$E0          ;EXECUTE
547            STA    $04           ;BFR N.4
548
549 * ATTENDE CHE LA ROUTINE DI IRQ.
550 * SVOLGA IL SUO LAVORO
551 LP11       LDA    $04
552            BMI    LP11
553
554 * ESCE SE IL CODICE DI ERRORE E'
555 * UGUALE A #$7F: "ERRORE" FLAG DI
556 * END OF FILE (E.O.F.)
557            CMP    #$7F
558            BEQ    LP13
559
560 * PROSEGUE SOLO SE L'ERRORE
561 * RISCONTRATO E' MINORE DI DUE
562            CMP    #$02
563            BCC    LP9
564
565 * DECREMENTA IL NUMERO DEI
566 * TENTATIVI E RIPROVA
567            DEC    MAXTENT
568            BPL    LP10
569
570 * SE I TENTATIVI SONO ESAURITI
571 * ALLORA PROVA AD ALLINEARE LA
572 * TESTINA E RITENTA LA LETTURA
573            LDY    MAXTENT
574            INY
575
576 * SALTA SE IL COMANDO DI "BUMP"
577 * (#$C0) E' GIA' STATO ESEGUITO
578            BNE    LP12
579
580 * COMANDO DI BUMP PER LA TESTINA
581            LDA    #$C0
582            JSR    $D58E
583            DEC    MAXTENT
584
585 * ESEGUE ULTERIORI TENTATIVI DOPO
586 * IL COMANDO DI BUMP
587 LP12       LDY    MAXTENT
588            CPY    #$FB
589            BCS    LP10
590
591 * SE FALLISCONO ANCORA DICHIARA
592 * ILLEGGIBILE IL SETTORE
593            LDA    #$60          ; RTS
594            STA    LAST-ORIGIN+$700
595
596            LDA    #$FF
597            JSR    ROUTINE-ORIGIN+$700
598
599 * SETTA IL TIMER 1
600            LDA    #$3A
```

```
601            STA    TIMER1
602
603 * ACC. - ERRORE ; X - N.RO BUFFER
604            LDA    $04
605            LDX    #$04
606            JMP    $E60A
607
608 ********************************
609 *    FINE DEL LOAD: TUTTO OK!   *
610 ********************************
611
612 * SETTA IL TIMER 1
613 LP13       LDA    #$3A
614            STA    TIMER1
615
616 * RICARICA LA B.A.M. NEL BUFFER 4
617            JMP    $D048
618
619 ********************************
620 * FINE DELLA ROUTINE DEL DRIVE *
621 ********************************
622
623 *------------------------------*
624
625 ********************************
626 * INIZIO DELLA ROUTINE DUPLOAD *
627 * NEL COMMODORE 64.            *
628 ********************************
629
630 * SALVA IL FLAG LOAD/VERIFY
631 SLOAD      STA    VERCK
632
633 * PERIFERICA <4 - LOAD NORMALE
634            LDA    FA
635            CMP    #$04
636            BCS    CHECK2
637
638 * CARICAMENTO NORMALE DEL FILE
639 NODUP      LDA    VERCK
640            JMP    ($7FE)
641
642 * SE NON C'E' IL NOME: ERRORE
643 CHECK2     LDA    FNLEN
644            BNE    CHECK3
645            RTS
646
647 * SE FILE-DIRECTORY: LOAD NORMALE
648 CHECK3     LDY    #$00
649            LDA    (FNADR),Y
650            CMP    #'$
651            BEQ    NODUP
652
653 * N.RO FILE LOGICO-DEVICE NUMBER
654            LDA    FA
655            STA    LA
656
657 * SCRIVE "SEARCHING FOR"...
658            JSR    PRLOAD
659
660 * RIPRISTINA L'INPUT/OUTPUT (I/O)
661            JSR    RSTIO
662
663 * SALVA INDIRIZZO SECONDARIO 0/1
664            LDX    SA
665            STX    TEMPX
666
667 * USA IL CANALE RISERVATO AL LOAD
668 * (LO 0) CON PREFISSO 6
669            LDA    #$60
670            STA    SA
671
672 * APRE FILE SUL DRIVE E SALTA IL
673 * CONTROLLO NELLA ROUTINE BOPEN
674 * SULLA CORRETTEZZA DEL DEVICE
675            JSR    BOPEN
676
677 * SI FA DARE LO STATO DAL DRIVE
678 * PER STABILIRE SE C'E' O MENO
679            LDA    FA
680            JSR    TALK
681            LDA    SA
682            JSR    SATALK
683            JSR    INBYTE
684
685 * ANALIZZA IL BIT 1 DELLO STATUS
686 * SE AD 1 - RITARDO TEMPO IN READ
687            LDA    ST
688            LSR
689            LSR
690            BCC    ZR176
691
692 * ?FILE NOT FOUND ERROR
693            JMP    FERROR
694
695 * SCRIVE "LOADING"/"VERIFYING"
696 ZR176      JSR    LDVER
697
698 * INIZIALIZZA IL CONTABLOCCHI
699            LDA    #'0
700            STA    CBLOCK
701            STA    CBLOCK+1
702            STA    CBLOCK+2
703
704 ********************************
705 * SCRIVE 256 DATI A PARTIRE DA *
706 * $0700 (LOCAZIONE DRIVE) IN   *
707 * BLOCCHI DA 32 VALORI L'UNO.  *
708 ********************************
709
710 * AZZERA IL CONTATORE $A4
711            LDA    #$00
712            STA    ZA4
713
714 ********************************
715 * M-W, [$A4], $07, $20, ....   *
716 * POKE1541 $07[$A4]; $20; DATI *
717 ********************************
718
719 * DICE AL DRIVE "M-"...
720 ZR180      JSR    MEMWR
```

```
721
722            LDA    #'w
723            JSR    OUTBYT
724            LDA    ZA4
725            JSR    OUTBYT
726            LDA    #$07
727            JSR    OUTBYT
728            LDA    #$20
729            JSR    OUTBYT
730
731 * CARICA IN Y [$A4], INCREMENTA
732 * $A4 DI $20 = 32 DECIMALE
733            LDY    ZA4
734            CLC
735            LDA    ZA4
736            ADC    #$20
737            STA    ZA4
738
739 * TRASFERISCE $20 BYTES AL DRIVE
740 ZR1A0      LDA    ORIGIN,Y
741            JSR    OUTBYT
742            INY
743            CPY    ZA4
744            BNE    ZR1A0
745
746 * FINE DEL BLOCCO DA $20 BYTES
747            JSR    ULSTEN
748
749 * SE NON SONO FINITI I $100 BYTES
750 * ALLORA RIPETE L'OPERAZIONE
751            LDA    ZA4
752            BNE    ZR180
753
754 **********************************
755 * MANDA IN ESECUZIONE IL L.M.    *
756 * NEL DRIVE: SYS1541 $07AB.      *
757 **********************************
758
759 * DICE AL DRIVE "M-"...
760            JSR    MEMWR
761
762            LDA    #'e
763            JSR    OUTBYT
764            LDA    #<EXEC-ORIGIN+$700
765            JSR    OUTBYT
766            LDA    #>EXEC-ORIGIN+$700
767            JSR    OUTBYT
768            JSR    ULSTEN
769
770 * SALVA STATO DEL BUS SERIALE
771            LDA    BUS
772            PHA
773
774 * SECONDO BYTE: BUS RESETTATO
775            AND    #$07
776            STA    ZA5
777
778 * PRIMO BYTE: CAMBIA SOLO "ATN"
779            ORA    #$08
780            STA    ZA4
781
782 * SALVA LO STATO DEGLI SPRITES
783            LDA    ENASPR
784            PHA
785
786 * CANCELLA SPRITES PERCHE' NON
787 * INTERFERISCANO CON "DUPBYTE"
788            LDA    #$00
789            STA    ENASPR
790
791 * SALVA STATO ROM, I/O, TAPE
792            LDA    R6510
793            PHA
794
795 * DISABILITA LE INTERRUZIONI
796            SEI
797
798 * PREDISPONE $01 PER SCRITTURA
799 * SU TUTTA LA RAM. ORA RIMANE
800 * SOLO L' I/O.
801            LDA    #$35
802            STA    R6510
803
804 **********************************
805 * X=PUNTATORE ALL'INTERNO DEL    *
806 * BLOCCO DA LEGGERE. I PRIMI 4   *
807 * BYTES PUNTANO AL SUCCESSIVO    *
808 * BLOCCO E INDIRIZZO DI LOAD     *
809 * QUINDI RESTANO 252=$FC BYTES   *
810 **********************************
811
812            LDX    #$FC
813
814 * SE TASTO RUN STOP E' PREMUTO
815 * INTERROMPE LA LETTURA DEL FILE
816 BREAD      LDA    #$7F
817            STA    PRA1
818            CMP    PRB1
819            BEQ    ERROR
820
821 * CARICA LA TRACCIA DEL BLOCCO
822 * SEGUENTE
823            JSR    DUPBYTE
824
825 * SE A=$FF LA TRACCIA NON ESISTE
826 * ALTRIMENTI CONTINUA LA LETTURA
827            CMP    #$FF
828            BNE    BREAD2
829
830 * SE A=$FF: ESCE CON L'ERRORE DI
831 * DEVICE TIME OUT (BIT 1 DI $90)
832 ERROR      LDA    #$02
833            STA    ST
834
835 * SALTO INCONDIZIONATO
836            BNE    EXIT
837
838 **********************************
839 * CARICA IL PROSSIMO BLOCCO.     *
840 **********************************
```

```
841
842 * SALVA LA TRACCIA
843 BREAD2   PHA
844
845 * LEGGE SETTORE/N.BYTES
846          JSR   DUPBYTE
847
848 * SALVA IL N.RO DI BYTES
849          TAY
850
851 * LEGGE LO STACK
852          PLA              ;TRACCIA
853          PHA              ;TRACCIA
854
855 * E' IL PRIMO BLOCCO?
856          CPX   #$FE
857
858 * SALVA LA RISPOSTA
859          PHP
860
861 * E' L'ULTIMO BLOCCO?
862          CMP   #$00
863          BNE   BREAD3
864
865 * SI: X = Y = NUMERO DI BYTES
866          TYA
867          TAX
868
869 * TOGLIE 2 BYTES DI TRK & SKT
870          DEX
871          DEX
872
873 * CONTROLLA SE E' IL PRIMO BLOCCO
874          PLP
875          PHP
876
877 * SE SI SOTTRAE 2 BYTES
878          BEQ   BREAD3
879          DEX
880          DEX
881
882 * E' IL PRIMO BLOCCO?
883 BREAD3   PLP
884
885 * SE SI SETTA L'INDIRIZZO
886          BEQ   BREADX
887
888 ********************************
889 * CARICAMENTO INDIRIZZO INIZIO *
890 * ,1 DA DRIVE; ,0 UTENTE ($C3) *
891 * INDIRIZZO MEMORIZZATO IN $AE *
892 ********************************
893
894          JSR   DUPBYTE
895          STA   ZAE
896          JSR   DUPBYTE
897          LDY   TEMPX
898          BNE   SETDRV
899          LDA   USER
900          STA   ZAE
901          LDA   USER+1
902 SETDRV   STA   ZAE+1
903
904 * CARICA X BYTES DAL BLOCCO DATI
905 BREADX   JSR   DUPBYTE
906
907 *********************************
908 * ROUTINE STOREA; LOAD X BYTES *
909 *********************************
910
911          DEC   R6510
912          LDY   #$00
913          CPY   VERCK
914          BNE   STORE2
915          STA   (ZAE),Y
916 STORE2   CMP   (ZAE),Y
917          BEQ   STORE3
918          LDA   #$10          ;ERRORE
919          ORA   ST
920          STA   ST
921 STORE3   INC   R6510
922
923          INC   ZAE
924          BNE   BREADX1
925          INC   ZAE+1
926 BREADX1  DEX
927          BNE   BREADX
928
929 ***** FINE ROUTINE BREAD X *****
930
931 *********************************
932 * INCREMENTA IL CONTABLOCCHI.  *
933 *********************************
934
935          LDA   CBLOCK+2
936          CMP   #'9
937          BEQ   INCDEC
938          INC   CBLOCK+2
939          BNE   ENDINC
940 INCDEC   LDA   #'0
941          STA   CBLOCK+2
942          LDA   CBLOCK+1
943          CMP   #'9
944          BEQ   INCCEN
945          INC   CBLOCK+1
946          BNE   ENDINC
947 INCCEN   LDA   #'0
948          STA   CBLOCK+1
949          INC   CBLOCK
950
951 *********************************
952 * CARICA IN X NUMERO BYTES DEL *
953 * PROSSIMO BLOCCO; 254 SE NON  *
954 * E' L'ULTIMO SETTORE.         *
955 *********************************
956
957 ENDINC   LDX   #$FE
958
959 * E' ULTIMO BLOCCO?
960          PLA              ;TRACCIA
```

```
961          BEQ   EXIT
962          JMP   BREAD
963
964 *********************************
965 * RIMETTE TUTTO COME PRIMA DEL *
966 * CARICAMENTO.                  *
967 *********************************
968
969 * SE NEL CARICAMENTO IL TURBO E'
970 * STATO DISABILITATO: RIABILITA
971 EXIT     LDA   ILOAD
972          CMP   $7FE
973          BNE   EXIT2
974          LDA   ILOAD+1
975          CMP   $7FF
976          BNE   EXIT2
977          JSR   ENABLE
978
979 * RIMETTE $01 COME PRIMA
980 EXIT2    PLA
981          STA   R6510
982
983 * RIABILITA LO STATO SPRITES
984          PLA
985          STA   ENASPR
986
987 * RIMETTE IL BUS COME PRIMA LOAD
988          PLA
989          STA   BUS
990
991 * CHIUDE IL CANALE COMUNICAZIONE
992          JSR   BCLOSE
993
994 * AGGÍUNGE "END OF FILE" A ST
995          LDA   #$40
996          ORA   ST
997          STA   ST
998
999 * PORTA IL BIT 2 DI ST NEL CARRY
1000         LSR
1001         LSR
1002
1003 * CONSERVA STATO DEL PROCESSORE
1004         PHP
1005
1006 * STAMPA GLI INDIRIZZI DEL LOAD
1007         JSR ADDRESS
1008
1009 * RECUPERA STATO DEL PROCESSORE
1010         PLP
1011
1012 * RICARICA IN X & Y L'INDIRIZZO
1013 * FINALE DEL CARICAMENTO
1014         JSR   IFINALE
1015
1016 * CARICA $1D=29 CRSR RIGHT IN A
1017         LDA   #$1D
1018
1019 * RIABILITA LE INTERRUZIONI
1020         CLI
1021
1022 *********************************
1023 * FINE DELLA ROUTINE DI LOAD.   *
1024 *********************************
1025
1026         RTS
1027
1028 *********************************
1029 *  ROUTINE DI STAMPA DEI LOAD   *
1030 *  ADDRESSES IN ESADECIMALE.    *
1031 *********************************
1032
1033 * CONTROLLA INDIRIZZO SECONDARIO
1034 * PER SAPERE DOVE TROVARE
1035 * L'INDIRIZZO INIZIALE DEL LOAD
1036 ADDRESS  LDA   TEMPX
1037         BNE   MEMREAD
1038
1039 * CARICA IL VALORE STANDARD
1040 * DELL'INDIRIZZO INIZIALE
1041         LDX   USER
1042         LDY   USER+1
1043         CLV
1044         BVC   INIZIO
1045
1046 * ESEGUE IL M-R DI $402 DEL 1541
1047 MEMREAD  JSR   MEMWR
1048         LDA   #'r
1049         JSR   OUTBYT
1050         LDA   #$02
1051         JSR   OUTBYT
1052         LDA   #$04
1053         JSR   OUTBYT
1054         LDA   #$02
1055         JSR   OUTBYT
1056         JSR   ULSTEN
1057
1058 * LEGGE I DUE BYTES LOW/HIGH
1059 * DELL'INDIRIZZO INIZIALE E LI
1060 * TRASFERISCE IN X & Y
1061         LDA   FA
1062         JSR   TALK
1063         LDA   #$6F
1064         JSR   SATALK
1065         JSR   INBYTE
1066         TAX
1067         JSR   INBYTE
1068         TAY
1069         JSR   UNTALK
1070
1071 * STAMPA L'INDIRIZZO INIZIALE
1072 INIZIO   JSR CONVERT1
1073
1074 * STAMPA L'INDIRIZZO FINALE
1075         JSR   IFINALE
1076
1077 * STAMPA LO SPAZIO ED IL "$"
1078 CONVERT1 LDA   #' '
1079         JSR   CHROUT
1080         LDA   #'$
```

```
1081          JSR    CHROUT
1082
1083 * STAMPA LA PARTE "HIGH"
1084          TYA
1085          JSR    CONVERT2
1086
1087 * STAMPA LA PARTE "LOW"
1088          TXA
1089
1090 ********************************
1091 *  ROUTINE DI CONVERSIONE IN    *
1092 *  ESADECIMALE E DI STAMPA.     *
1093 ********************************
1094
1095 * CONSERVA IL VALORE
1096 CONVERT2 PHA
1097
1098 * STAMPA IL NIBBLE ALTO
1099          AND    #$F0
1100          JSR    HIGH
1101
1102 * RECUPERA IL VALORE
1103          PLA
1104
1105 * STAMPA IL NIBBLE BASSO
1106          AND    #$0F
1107          BPL    LOW
1108
1109 * METTE IN "LOW" IL NIBBLE "HIGH"
1110 HIGH     LSR
1111          LSR
1112          LSR
1113          LSR
1114
1115 * STAMPA IL NIBBLE LOW
1116 LOW      TAY
1117          CLC
1118          ADC    #'0
1119          CPY    #$0A
1120          BCC    PRINT
1121          CLC
1122          ADC    #$07
1123 PRINT    JMP    CHROUT
1124
1125 ********************************
1126 * RIEMPITIVI PER LE TABELLE DI *
1127 * CARICAMENTO BYTES DAL DRIVE. *
1128 ********************************
1129
1130          NOP
1131          NOP
1132          NOP
1133          NOP
1134          NOP
1135          NOP
1136          NOP
1137          NOP
1138          NOP
1139          NOP
1140          NOP
1141          NOP
1142          NOP
1143
1144 ********************************
1145 * TABELLE PER CARICAMENTO BYTE *
1146 *------------------------------*
1147 * QUESTO INDIRIZZO DEVE ESSERE *
1148 * NELLA FORMA $XX00 CIOE' DEVE *
1149 * TERMINARE CON DUE ZERI (HEX) *
1150 ********************************
1151
1152 TAB1      HEX    A0,A0,A0,A0 ;TAB 1.0
1153           HEX    A0,A0,A0,A0
1154 TAB2      HEX    50,50,50,50 ;TAB 2.0
1155           HEX    50,50,50,50
1156 TAB3      HEX    0A,0A,0A,0A ;TAB 3.0
1157           HEX    0A,0A,0A,0A
1158 TAB4      HEX    05,05,05,05 ;TAB 4.0
1159           HEX    05,05,05,05
1160
1161 ********************************
1162 * PRENDE UN BYTE DAL FAST 1541 *
1163 ********************************
1164
1165 DUPBYTE   LDA    ZA4
1166           STA    BUS
1167 NOREADY   LDA    BUS
1168           BPL    NOREADY
1169 VICDMA    LDA    RASLIN
1170           CMP    #$31
1171           BCC    OK
1172           AND    #$06
1173           CMP    #$02
1174           BEQ    VICDMA
1175 OK        LDA    ZA5
1176           STA    BUS
1177
1178 ********************************
1179 * ATTESA PER 20 CICLI DI CLOCK *
1180 ********************************
1181
1182           JSR    PART2      ; 6*T
1183 NULL      RTS
1184
1185 ********************************
1186 *          TABELLA 1           *
1187 ********************************
1188
1189           HEX    20,20,20,20 ;TAB 1.1
1190           HEX    20,20,20,20
1191           HEX    10,10,10,10 ;TAB 2.1
1192           HEX    10,10,10,10
1193           HEX    02,02,02,02 ;TAB 3.1
1194           HEX    02,02,02,02
1195           HEX    01,01,01,01 ;TAB 4.1
1196           HEX    01,01,01,01
1197
1198 ********************************
1199 * ROUTINE DI DELAY PER DUPBYTE *
1200 ********************************
```

```
1201
1202 PART2      NOP                    ; 2*T
1203            ROL    NULL            ; 6*T
1204            ROR    NULL            ; 6*T
1205            LDY    BUS
1206            LDA    TAB1,Y
1207            LDY    BUS
1208            ORA    TAB2,Y
1209            LDY    BUS
1210            ORA    TAB3,Y
1211            LDY    BUS
1212            ORA    TAB4,Y
1213            RTS
1214
1215 *********************************
1216 *          TABELLA 2            *
1217 *********************************
1218
1219            HEX    80,80,80,80 ;TAB 1.2
1220            HEX    80,80,80,80
1221            HEX    40,40,40,40 ;TAB 2.2
1222            HEX    40,40,40,40
1223            HEX    08,08,08,08 ;TAB 3.2
1224            HEX    08,08,08,08
1225            HEX    04,04,04,04 ;TAB 4.2
1226            HEX    04,04,04,04
1227
1228 *********************************
1229 * INVIA AL DRIVE LISTEN+"M-"...*
1230 *********************************
1231
1232 MEMWR      LDA    FA
1233            JSR    LISTEN
1234            LDA    #$6F
1235            JSR    SALIST
1236            LDA    #'m
1237            JSR    OUTBYT
1238            LDA    #'-
1239            JMP    OUTBYT
1240
1241 *********************************
1242 * CARICA X & Y CON L'INDIRIZZO *
1243 * FINALE DEL CARICAMENTO.       *
1244 *********************************
1245
1246 IFINALE    LDX    ZAE
1247            LDY    ZAE+1
1248            RTS
1249
1250 *********************************
1251 *          SPAZIO LIBERO        *
1252 *********************************
1253
1254            NOP
1255            NOP
1256            NOP
1257            NOP
1258            NOP
1259            NOP
1260            NOP
1261
1262 *********************************
1263 *          TABELLA 3            *
1264 *********************************
1265
1266            HEX    00,00,00,00 ;TAB 1.3
1267            HEX    00,00,00,00
1268            HEX    00,00,00,00 ;TAB 2.3
1269            HEX    00,00,00,00
1270            HEX    00,00,00,00 ;TAB 3.3
1271            HEX    00,00,00,00
1272            HEX    00,00,00,00 ;TAB 4.3
1273            HEX    00,00,00,00
1274
1275 FINISH
1276
1277            ORG    $010A
1278
1279 *********************************
1280 * ROUTINE DI INTERPRETAZIONE    *
1281 * DELL'EVENTUALE COMANDO "←".   *
1282 *********************************
1283
1284 ISTACK
1285
1286 * CONSERVA L'ATTUALE PUNTATORE
     NEL BUFFER DEI COMANDI
1287 EXESTA     LDA    $7A
1288            PHA
1289            LDA    $7B
1290            PHA
1291
1292 * PRENDE IL SUCCESSIVO CARATTERE
1293            JSR    $0073
1294
1295 * SALTA SE E' UGUALE AD "←"
1296            CMP    #'←
1297            BEQ    INVERT
1298
1299 * SE E' DIFFERENTE DA "←"
     RIPRISTINA IL PUNTATORE ORIGINALE
1300            PLA
1301            STA    $7B
1302            PLA
1303            STA    $7A
1304
1305 * SALTA ALL'"IGONE" ORIGINALE
1306            JMP    $A7E4
1307
1308 *********************************
1309 * MODIFICA I VALORI DI "PHA"    *
1310 * DEL PUNTATORE $7A/$7B CON     *
1311 * L'INDIRIZZO DI RITORNO $A7AE *
1312 * PER L'ESECUZIONE DELL'ISTRU-  *
1313 * ZIONE SUCCESSIVA.             *
1314 *********************************
1315
1316 INVERT     TSX
1317            LDA    #$A7
```

```
1318          STA    STACK+2,X
1319          LDA    #$AE-1
1320          STA    STACK+1,X
1321
1322 *********************************
1323 * AZZERA I CODICI NEL BUFFER    *
1324 * DEI COMANDI PER EVITARNE      *
1325 * ULTERIORI INTRPRETAZIONI.     *
1326 *********************************
1327
1328          LDA    #$00
1329          TAY
1330          STA    ($7A),Y
1331          INY
1332          INY
1333          STA    ($7A),Y
1334
1335 *********************************
1336 * "INVERTE" LO STATO DI ON/OFF *
1337 * DELLO SPEEDISK.               *
1338 *********************************
1339
1340          LDA    ILOAD+1
1341          CMP    #>ENTRY
1342          BNE    ENABLEM
1343
1344 *********************************
1345 * DISABILITA LO SPEEDISK.       *
1346 *********************************
1347
1348 DISABLEM CLC
1349
1350 * BYTE DI COPERTURA
1351          HEX    24
1352
1353 *********************************
1354 * ABILITA LO SPEEDISK.          *
1355 *********************************
1356
1357 ENABLEM  SEC
1358
1359 * CONSERVA IL FLAG ON/OFF
1360          PHP
1361
1362 *********************************
1363 * STAMPA MESSAGGIO DI ON/OFF.   *
1364 *********************************
1365
1366          LDX    #ENDMSG-MSGONOFF
1367 ONOFF    LDA    MSGONOFF,X
1368          JSR    CHROUT
1369          DEX
1370          BPL    ONOFF
1371
1372 * SCRIVE "ON"
1373          LDA    #'n
1374          PLP
1375          PHP
1376          BCS    ON
1377
1378 * SCRIVE "OFF"
1379          LDA    #'f
1380          JSR    CHROUT
1381 ON       JSR    CHROUT
1382          LDA    #$0D
1383          JSR    CHROUT
1384          PLP
1385          BCC    DISABLE
1386
1387 *********************************
1388 * ABILITA LO SPEEDISK SUL C/64 *
1389 *********************************
1390
1391 ENABLE   LDX    #<ENTRY
1392          LDY    #>ENTRY
1393          CLV
1394          BVC    ALTER
1395
1396 *********************************
1397 * DISABILITA LO SPEEDISK.       *
1398 *********************************
1399
1400 DISABLE  LDX    $7FE
1401          LDY    $7FF
1402 ALTER    STX    ILOAD
1403          STY    ILOAD+1
1404          RTS
1405
1406 *********************************
1407 * MESSAGGI DI DIS/ABILITAZIONE *
1408 * DELLO SPEEDISK SUL C/64.      *
1409 *********************************
1410
1411 MSGONOFF REV    'speedisk o'
1412 ENDMSG   HEX    91           ; CRSR UP
1413
1414 *********************************
1415 * NUOVA ROUTINE DI N.M.I. IRQ. *
1416 *********************************
1417
1418 * CONSERVA NELLO STACK I VALORI
     DI "A", "X" ED "Y".
1419 NEWNMI   PHA
1420          TXA
1421          PHA
1422          TYA
1423          PHA
1424
1425 * ESEGUE UN CLEAR SU N.M.I.
1426          LDA    #$7F
1427          STA    $DD0D
1428
1429 * LETTURA E CLEAR DEI FLAGS
1430          LDA    $DD0D
1431
1432 * PROSEGUE SE LA RS-232 NON E' ATTIVA
1433          BPL    NOIRQ
1434
1435 * ALTRIMENTI ESEGUE IL N.M.I. INTERNO
1436 IRQ      JMP    $FE72
```

```
1437
1438 * SCANDISCE LA TASTIERA
1439 NOIRQ    JSR    $F6BC
1440
1441 * CONTROLLA L'EVENTUALE PRESSIONE
      DEL TASTO DI RUN/STOP
1442          JSR    $FFE1
1443
1444 * SALTA SE IL R/S NON E' PREMUTO
1445          BNE    IRQ
1446
1447 * INIZIALIZZA I VETTORI I/O
1448          JSR    $FD15
1449
1450 * INIZIALIZZA LE PERIFERICHE I/O
1451          JSR    $FDA3
1452
1453 * INIZIALIZZA SCHERMO E TASTIERA
1454          JSR    $E518
1455
1456 * RISTABILISCE I COLORI DI DEFAULT
      DELLO SPEEDISK
1457          LDA    #$0B
1458          STA    $D020
1459          LDA    #$00
1460          STA    $D021
1461          LDA    #$9B
1462          JSR    SAVEBYTS
1463
1464 * RIPRISTINA IL VETTORE DI N.M.I.
      ALLA PRESENTE ROUTINE
1465          LDX    #<NEWNMI
1466          LDY    #>NEWNMI
1467          STX    NMINV
1468          STY    NMINV+1
1469
1470 * PARTENZA A CALDO PER IL BASIC
1471          JMP    ($A002)
1472
1473 FSTACK
1474
1475          ORG    $07E8
1476
1477 *********************************
1478 * INGRESSO PER LO SPEEDLOAD.    *
1479 *********************************
1480
1481 ISCR
1482
1483 * CONSERVA IL FLAG LOAD/VERIFY IN X
1484 ENTRY    TAX
1485
1486 * CONSERVA LO STATO RAM/ROM DEL C/64
1487          LDA    R6510
1488          PHA
1489
1490 * BASIC OFF PER ACCEDERE ALLO
      SPEEDISK DA $A000
1491          LDA    #$36
1492          STA    R6510
1493
1494 * RIPRISTINA IL FLAG L/V
1495          TXA
1496
1497 * ESEGUE LO "SPEEDISK-LOAD"
1498          JSR    SLOAD
1499
1500 * RIPRISTINA LO STATO RAM/ROM
1501          PLA
1502          STA    R6510
1503
1504 * FINE DEL LOAD
1505          RTS
1506
1507 *********************************
1508 * ROUTINE DI "RISPARMIO" BYTES *
1509 * DELLO STACK, POSIZIONATA NEI *
1510 * BYTES LIBERI RIMANENTI PRIMA *
1511 * DELL'INIZIO BASIC $801.      *

1512 *********************************
1513
1514 SAVEBYTS JSR    CHROUT
1515          JMP    ENABLEM
1516
1517 FSCR
1518
1519          LST    ON
1520
1521          END
```

```
10 rem controllo write-protect
20 p=peek(186): if p<4 then p=8
30 open 1,p,15: close 1
40 if st then a$="drive spento!": goto 90
50 open 1,p,15,"m-r"+chr$(0)+chr$(28)
60 get# 1,a$: a=asc(a$+chr$(0)) and 16
70 if a=0 then a$="tacca presente!"
80 if a=16 then a$="tacca assente."
90 print a$: close 1: end: rem return

ready.
```

```
1 open 8,8,8,"nome": for k=0 to 1: get# 8,a$: k=st: print a$;: next k: close 8

ready.
```

```
10 rem error channel
20 for l=49152 to 49263: read d
30 poke l,d: c=c+d: next l
40 if c<>14986 then print "errore!": stop
50 sys 49152: print "sqattivato.": new
60 data 169,11,141,2,3,169,192,141,3,3,96,56,32,240,255,142,112,192
70 data 140,113,192,169,19,32,210,255,169,18,32,210,255,160,39,169,160,153
80 data 0,4,173,134,2,153,0,216,136,208,242,169,62,32,210,255,169,15
90 data 162,8,160,15,32,186,255,169,0,32,189,255,32,192,255,162,15,32
100 data 198,255,32,207,255,201,13,240,5,32,210,255,208,244,174,112,192,172
110 data 113,192,24,32,240,255,169,146,32,210,255,169,15,32,195,255,32,204
120 data 255,76,131,164

ready.
```

```
10 rem c=64 directory array
20 for a=828 to 1006: read b: poke a,b
30 c=c+b: next a: if c=22602 then 150
40 print "errore nei data!": stop
50 data 169,14,162,8,160,0,32,186,255,169,4,162,235,160,3,32,189,255,32,192
60 data 255,162,14,32,198,255,160,0,140,57,3,132,250,140,56,3,165,48,133
70 data 254,24,165,47,105,7,144,2,230,254,133,253,169,0,133,251,169,192,133
80 data 252,160,1,32,207,255,166,144,240,15,32,204,255,169,14,32,195,255
90 data 172,56,3,136,132,254,96,192,6,144,45,201,34,208,10,173,57,3,73,1
100 data 141,57,3,16,31,174,57,3,208,30,192,32,208,22,160,0,165,250,145,253
110 data 132,250,24,165,253,105,3,144,2,230,254,133,253,76,119,3,200,76,121
120 data 3,140,59,3,160,0,145,251,165,250,208,14,160,1,165,251,145,253,200
130 data 165,252,145,253,238,56,3,230,250,172,59,3,230,251,208,2,230,252,76
140 data 166,3,36,48,58,42
150 dim a$(144):sys 828
160 e=peek(254): for c=1 to e
170 print a$(c): next c: end

ready.
```

```
ready.



10 open 1,8,15,"i0":open 8,8,0,"$0:"
20 for j=1 to 18:get# 8,a$,b$
30 c$=c$+a$+b$: next j: close 8
40 b=asc(a$+chr$(0))+256*asc(b$+chr$(0))
50 print "disco: ";mid$(c$,7,16)
60 print b;"blocchi liberi": end

ready.


10 rem legge directory
20 open 8,8,0,"$": l=8192
30 get# 8,a$: poke l,asc(a$+chr$(0))
40 l=l+1: if st=0 then 30
50 close8: print st, l: end

ready.



10 rem basic directory
20 open 8,8,0,"$": get# 8,a$,a$
30 get# 8,a$,a$: if a$="" then close 8: end
40 get# 8,b$,l$
50 ln$=str$(asc(b$+chr$(0))+asc(l$+chr$(0))*256): print ln$;" ";
60 get# 8,a$: printa$;: if a$<>"" then 60
70 print: goto 30

ready.



10 rem 64 getspeed - run aug.87 pg.14
20 for a=820 to a+57: read b: poke a,b: next a
30 data 169,5,162,241,160,3,32,189,255,169,3,162,8,160,3,32,186,255,32,192
40 data 255,162,3,32,198,255,32,159,255,201,32,240,16,32,183,255,201,64,240,9
50 data 32,207,255,32,210,255,76,78,3,169,3,32,195,255,32,204,255,96
60 input "File seq. da leggere";a$
70 for x=1 to len(a$): poke 1008+x,asc(mid$(a$,x,1)): next x
80 poke 821,len(a$): sys 820: end

ready.


100 REM LEGGE E VISUALIZZA UN BLOCCO
110 :
120 PRINT CHR$(147)
130 INPUT "TRACCIA DA LEGGERE";T
140 INPUT "SETTORE DA LEGGERE";S
150 OPEN 1,8,15,"I"
160 OPEN 2,8,2,"#"
170 PRINT# 1,"U1";2;0;T;S
```

```
180 FOR K=0 TO 255
190 GET# 2,A$
200 X=ASC(A$+CHR$(0))
210 PRINT X;
220 IF X>32 AND X<128 THEN PRINT CHR$(X);
230 PRINT ,
240 NEXT K
250 CLOSE 2
260 CLOSE 1
270 END

READY.
```

```
100 REM: DISK STATUS PER 1541
110 REM: (@) 1988 BY A. DIANO
120 :
130 DATA 169,0,141,32,208,141,33,208
140 DATA 169,155,32,210,255,169,147,32
150 DATA 210,255,174,2,3,172,3,3
160 DATA 134,178,132,179,165,20,24,105
170 DATA 44,141,2,3,165,21,105,0
180 DATA 141,3,3,96,8,72,138,72
190 DATA 152,72,56,32,240,255,138,72
200 DATA 152,72,169,19,32,210,255,169
210 DATA 18,32,210,255,169,62,32,210
220 DATA 255,162,36,172,134,2,169,160
230 DATA 157,3,4,152,157,3,216,202
240 DATA 208,244,138,133,144,166,186,32
250 DATA 205,189,169,58,32,210,255,169
260 DATA 160,32,210,255,165,186,32,177
270 DATA 255,169,111,32,147,255,32,174
280 DATA 255,165,144,16,22,162,0,189
290 DATA 208,161,41,127,32,210,255,232
300 DATA 224,18,208,243,169,146,32,210
310 DATA 255,144,23,165,186,32,180,255
320 DATA 169,111,32,150,255,32,165,255
330 DATA 32,210,255,201,13,208,246,32
340 DATA 171,255,104,168,104,170,24,32
350 DATA 240,255,104,168,104,170,104,40
360 DATA 108,178,0
370 :
380 K=837: FOR A=0 TO 186: READ B
390 POKE K+A,B: C=C+B: NEXT A
400 IF C=24445 THEN SYS (K): NEW
410 PRINT CHR$(147);"ERRORE NEI DATA!"
420 STOP

READY.
```

```
100 gosub 560: rem questa dev'essere la prima linea del listato: non spostarla!
110 rem elimina errori d.o.s.
120 rem (@) 1988 - alex diano
130 rem per c64 & 1541 / 1571
140 poke 53280,11: poke 53281,0: poke 53282,4: poke 53283,11: poke 53284,0
150 for k=0 to 39: w$=w$+chr$(173): next k: s=54278: poke 198,0: poke 808,234
160 rem definizione parametri & colori
170 dim k(7): k(1)=11: k(2)=12: k(3)=15: k(4)=1: k(5)=15: k(6)=12: k(7)=11
180 b$=chr$(5): c$=chr$(159): g$=chr$(158): r$=chr$(150): q$=chr$(13)
```

```
190 gm$=chr$(152): v$=chr$(153): a$=chr$(154): gc$=chr$(155): s$=chr$(145)
200 n$=chr$(144): i$=chr$(18): poke 53265,peek(53265) or 64
210 print chr$(147);chr$(8);chr$(142);: gosub 500
220 print i$;g$;" -=*";b$;" elimina errori ";
230 print gc$;" cynu$ [@] 1988 ";g$;"*=- ";gm$;
240 for k=0 to 39: print chr$(45);: next k: print q$
250 print a$;"traccia formattata senza errori  ";g$;"18    ";b$;
260 for k=0 to 7: print chr$(157);: next k
270 input x$: tf=val(x$): if tf<1 or tf>35 then print s$;: goto 250
280 print q$;g$;"traccia da riparare";: for k=0 to 19: print chr$(32);: next k
290 print b$;: for k=0 to 19: print chr$(157);: next k: x$=""
300 input x$: tr=val(x$): if tr<1 or tr>35 then print s$;s$;: goto 280
310 print q$;q$;i$;w$;n$;" METTERE NEL DRIVE IL DISCO DA RIPARARE ";
320 print " E PREMERE LA BARRA SPAZIO PER INIZIARE ";b$;w$: poke 198,0: k=0
330 k=k+1: k=k+(k>7)*8: poke 53284,k(k)
340 for x=0 to 15: get x$: if x$=chr$(32) then poke 53284,0: x=15
350 next x: if x$<>chr$(32) then 330
360 gosub 500: open 8,8,8: close 8: if st=0 then 390
370 print s$;s$;s$;s$;r$;"    errore sul bus seriale: st = ";b$;st
380 print b$;"    PREMERE UN TASTO PER CONTINUARE      ": wait 198,1: run
390 print s$;s$;s$;s$;v$;" leggo, formatto, & riscrivo la traccia ";
400 print c$;" attendere anche se il led e' spento..."
410 open 8,8,15,"i": close 8: gosub 530: poke 2,tf: poke 3,tr: k=0: sys 49152
420 gosub 530: if tr=18 then print q$;gc$;"il disco necessita di una nuova b.a.m
.!"
430 print q$;q$;gc$;left$(w$,26);b$;q$;" ALTRE TRACCE DA RIPARARE ";
440 print g$;" (";gc$;"S";g$;"/";gc$;"N";g$;")";b$;"?";q$;gc$;left$(w$,26)
450 get x$: if x$=chr$(83) then run
460 if x$<>chr$(78) then 450
470 rem fine del programma
480 print chr$(155);chr$(147);chr$(9): poke 53265,peek(53265) and 191: end
490 rem subroutine sonora
500 poke s+18,21: poke s-1,9: poke s,0: poke s-5,48
510 poke s-2,32: pokes-2,33: return
520 rem controlla lo stato del drive
530 open 8,8,15: input# 8,x,y$,j,k: close 8: if x=0 then return
540 gosub 500: print q$;q$;i$;r$;"errore disco:";b$;x;y$;j;k;q$;q$: goto 380
550 rem lettura linee data
560 print chr$(147);tab(41);"lettura delle linee data: attendere...";chr$(13)
570 l=49152: for x=0 to 43: t=0: for y=0 to 15
580 read a: if a<0 or a>255 then 610
590 poke l,a: t=t+a: l=l+1: next y
600 read a: if t=a then 620
610 print chr$(13);"errore in linea";1000+x*10;"checksum <>";a: stop
620 print 1000+x*10;"ok",: next x: poke 2053,143: return
630 rem dati del riparatore in l.m.
1000 data 169,8,166,2,164,3,133,186,142,41,193,140,42,193,169,4, 1755
1010 data 162,41,160,193,32,194,192,169,5,162,197,160,193,32,194,192, 2278
1020 data 162,2,160,4,32,247,192,32,185,192,169,0,133,253,162,44, 1969
1030 data 160,4,32,247,192,32,148,192,240,9,32,91,192,230,253,230, 2284
1040 data 252,208,235,162,103,160,4,32,247,192,32,185,192,165,253,240, 2662
1050 data 9,32,132,192,198,253,230,252,208,243,96,32,21,193,160,0, 2251
1060 data 185,126,192,32,168,255,200,192,6,208,245,32,174,255,32,31, 2333
1070 data 193,160,0,32,165,255,145,251,200,208,248,76,171,255,77,45, 2481
1080 data 82,0,3,0,169,3,166,251,164,252,32,194,192,162,121,160, 1951
1090 data 4,76,247,192,32,21,193,160,0,185,179,192,32,168,255,200, 2136
1100 data 192,6,208,245,32,174,255,32,31,193,32,165,255,72,32,171, 2095
1110 data 255,104,96,77,45,82,9,0,1,169,0,133,251,169,96,133, 1620
1120 data 252,96,141,245,192,134,251,132,252,160,0,140,244,192,32,21, 2484
1130 data 193,162,0,189,241,192,32,168,255,232,224,6,208,245,162,32, 2541
1140 data 177,251,32,168,255,200,202,208,247,32,174,255,192,0,208,219, 2820
```

IN EDICOLA

N. 8 - LIRE 12.000
Commodore 64 Club
• Cover
• Modulus
• Canals of Mars
• Tetrix
• Golden Age
• Notepad
• Mission Twain
systems